NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

MYKYTA DEYNEHA

BACHELOR IN ELECTRICAL AND COMPUTER ENGINEERING

SMART SHOPPING IN RETAIL USING BEACON TECHNOLOGY

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

SMART SHOPPING IN RETAIL USING BEACON TECHNOLOGY

MYKYTA DEYNEHA

BACHELOR IN ELECTRICAL AND COMPUTER ENGINEERING

Adviser: João Rosas
Assistant Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2022

Smart Shopping in Retail Using Beacon Technology

"Our thought, our song
Will not die, will not perish...
Here, people, is our glory,
The glory of Ukraine!"

Taras Shevchenko

Acknowledgements

I want to express my gratitude and appreciation to my supervisor, João Rosas, for his support and guidance during the writing of my thesis and also for being one of the most influential professors who instigated my curiosity in pursuing and learning more about programming. For it became a true passion of mine.

I am deeply grateful to Nova School of Science and Technology for all these years of education, for giving me unforgettable academic experiences, for letting me meet exceptional people and for all the joyful moments that will live forever in my heart. The lectures and professors I had at this university helped shape me into the mature and experienced engineer I am today. For all this, I am truly grateful.

I want to express my gratefulness and indebtedness to my family, especially to my parents, my brother and my grandmother, Tamila Deyneha, Hennadiy Deyneha, Daniel Deyneha, Ganna Panko. Without their continuous support and guidance throughout my life, nothing of all my achievements would be possible. I could not ever fully express through this text my forever gratefulness for the care, patience and support my family has provided me. My family has always commented on how skilful and intuitive I was working with my computer and how well I was with technology and science, always commenting on me becoming one in my adult life.

I am grateful for the closest friends I have known before and during my university life. I want to give special thanks to João Marafuz, Afonso Batista, Dmytro Yaryna, Molly Bixby, Ana Sofia, João Ribeiro, João David, Martim Nogueira, Alexandre Goulão, João Goulão, João Castilho, Alexandra Reis, Leonor Garrido, Sergiu Nica, Diogo Noite, Daniel Feiteira, Serafim Ciobanu, Anastasiia Keba and to many whose name I did not mention here. Thank you for all our life experiences, parties, drinks we have shared and for all the advice and support you provided me throughout my life. I hope I have helped you as much as you have helped me, my sincere gratitude.

Abstract

The growth of the Internet of Things has been revolutionising several industries which have been implementing distinct IoT Systems to automate processes and operations. The retail industry has also been introducing automated systems such as automated stores. However, there is a great opportunity for innovation with the wide existence of technologies that can be potentially integrated into this type of system. The selection of technologies is an important aspect to take into account as energetic consumption and the complexity of the system are fully dependent on the technologies selected. Efficiency can be significantly impacted by the complexity of the system and provide difficulty in preserving the functioning of the system. Automated stores seek to improve the customer experience while collecting data on customer activities inside the store to benefit and grow the retail business. The thesis seeks to answer this problem by presenting a system for an automated retail shop that integrates beacon technology. Beacon technology uses Bluetooth Low Energy and implementing this type of device will maintain the low complexity of the system whilst taking into account the energy consumption. In the proposed system, these devices are used to collect the indoor location of the customers inside the shop to present the corresponding products in the location of the user. The products will be presented on a smartphone application interface where the user performs the shopping activity. Personnel from the store will also have an interface to track customers' activity and general events in the shop. The evaluation and testing of the system produced positive results, with acceptable accuracy in indoor localization and efficient functioning of the system applications which showed that we have a suitable solution for an automated retail shop.

Keywords: Internet of Things, Bluetooth Low Energy, Beacon Technology, Automated Retail Shop, Indoor Localization.

Resumo

O crescimento de Internet of Things tem vindo a revolucionar várias indústrias que têm vindo a implementar sistemas de IoT distintos para automatizar processos e operações. A indústria retalhista tem também vindo a introduzir sistemas automatizados, tais como lojas automáticas. No entanto, existe uma grande oportunidade de inovação com a ampla existência de tecnologias que podem ser potencialmente integradas neste tipo de sistemas. A selecção de tecnologias é um aspecto importante a ter em conta, uma vez que o consumo energético e a complexidade do sistema dependem totalmente das tecnologias integradas. A eficiência pode ser significativamente afectada pela complexidade do sistema e proporcionar dificuldade em preservar o funcionamento do sistema. As lojas automatizadas procuram melhorar a experiência do cliente enquanto recolhem dados sobre as actividades dos mesmos dentro da loja para beneficiar e crescer o negócio de retalho. A tese procura responder a este problema apresentando um sistema para uma loja automatizada de retalho que integra a tecnologia beacon. A tecnologia beacon utiliza Bluetooth Low Energy sendo que a implementação deste tipo de dispositivos manterá a baixa complexidade do sistema tendo em conta o consumo energético. No sistema proposto, estes dispositivos são utilizados para recolher a localização dos clientes no interior da loja para apresentar os produtos correspondentes que se encontram no local do utilizador. Os produtos serão apresentados numa interface de aplicação smartphone onde o utilizador realiza a sua actividade de compra. Os trabalhadores da loja teram também uma interface para acompanhar a actividade dos clientes e eventos gerais na loja. A avaliação e teste do sistema produziram resultados positivos, com uma precisão aceitável no calculo da localização e funcionamento eficiente das aplicações do sistema, o que demonstrou que temos uma solução aplicável para uma loja de retalho automatizada.

Palavras-chave: Internet of Things, Bluetooth Low Energy, Tecnologia Beacon, Loja de Retalho Automática, Localização Interior.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AES Advanced Encryption Standard 6
API Application Programming Interface 11, 12, 38, 39, 63

BLE Bluetooth Low Energy xi, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 28, 75, 85

CCM Cipher Block Chaining - Message Authentication Code 6
CDF Cumulative Distribution Function 8
CL Centroid Localization 19, 21
CSS Cascading Style Sheets 55
CVV Card Verification Value 52

FSPL free-space path loss phenomenon 16

GPS Global Positioning System 7, 13

IoT Internet of Things 1, 2, 6
IPSP Internet Protocol Support Profile 6
ISM Industrial, Scientific and Medical 6

LoRa Long Range 14

MVVM Model-View-ViewModel 47

NFC Near Field Communication 8

PDF Portable Document Format 53

QR Quick Response 8, 13

REST Representational State Transfer 25, 26, 38, 39, 63
RFID Radio Frequency Identification 7, 14
RSS Received Signal Strength 8, 17, 24, 25
RSSI Received Signal Strength Indicator 7, 12, 14, 15, 16, 17, 18, 19, 20, 67, 68, 76

SA smartphone application 24
SIG Special Interest Group 5, 6

ToA Time of Arrival 17

UI User Interface 57
URI Uniform Resource Identifier 25, 63
URL Uniform Resource Locator 13

WCL Weighted Centroid Localization 19, 20, 21
WCWCL Compensated Weighted Centroid Localization 20, 21
WWDC World Wide Developer Conference 11

1

Introduction

This chapter introduces the context involving automated systems in the retail industry. The motivation and objectives for the subject are also discussed.

1.1 Context

With the rapid growth of the Internet of Things (IoT) devices - predicted to reach around 75 billion connected devices by 2025 [72] - came several cutting-edge application systems. The new technological advances in IoT devices made a great contribution to the development of automated systems, integrating devices with sensing, identification, processing, communication, and networking functions. There have been presented several automated systems in the automotive, electronics manufacturing, medical, food service industries, etc. These systems proved to enhance the industries in performance, productivity, and lowering costs. Today in the retail industry specifically, customer service expectancy has risen as more retailers started to offer self-service and automated systems. These innovations have proven to positively affect organizations. Retail Industries benefit from strategies based on information by gathering and analysing relevant data in the market. In a world where competition is always increasing, these strategies can really differentiate the retailer from its competitors by satisfying the customer and gaining their loyalty. It is important for retailers to understand consumers' buying behaviours to build their strategies. Shopping is a daily activity performed regularly by countless people who behave differently depending on the type of shopping, the type of stores, and the products [73]. However, designing and implementing an automated system is not an easy task and requires different types of technologies.

1.2 Motivation

When Bluetooth Low Energy (BLE) technology was introduced, with it emerged devices with short-range wireless capability but with low power consumption, low latency, and

low complexity in communication, a proficient alternative to some of the previously existing wireless communication devices deployed in applications for retail, advertising, industrial settings, etc. An example of these devices is the BLE Beacons. With the release of Bluetooth 4.0, BLE beacons gained their place among one of the most used types of devices in IoT solutions due to their ease of deployment and performance in indoor positioning and proximity detection [78]. In combination with smartphone devices, beacon technology can be a feasible solution, since Bluetooth is integrated in android and apple smartphones, allowing communication with beacons. By using BLE beacon technology in an automated shopping system, a solution with minimal energy consumption can be developed because of the low-power consumption of beacons. Also, by collecting indoor positioning information as well as proximity detection information from the received signals on the smartphone from beacons, retailers can make use of collected data to understand costumer's practices and build retail strategies to further improve consumers' shopping experience, increasing their productivity and benefit from their use as well.

1.3 Problem Statement

There are big retail companies like Amazon which deployed their IoT grocery stores called Amazon Go, integrated with devices like cameras, pressure, infrared sensors, etc. By using Just Walk Out Technology, Amazon Go stores can track their customers with artificial intelligence during their shopping activities without interacting with any store employees at the checkout. How the system works is, a customer enters the grocery store, picks up his goods, and on leave, the system proceeds to link the products with the customer's account, billing them automatically [76]. Despite the efficiency of an automated system like Amazon implemented, most solutions that exist today are complex, some with high energy consumption and some propose solutions integrated with many different technologies where fixing and calibrating becomes too time-consuming. One of the main issues with retail not implementing new technologies like self-service and automated services is the idea of demanding the complexity of these systems. The challenge is to build a feasible solution, a practical automated shopping system which takes into account energy consumption and targets simplicity while efficiently improving consumers' shopping experience and giving retailers the ability to gather data to deploy effective retail strategies.

1.4 Objectives

Considering the implementation of an automated retail shop, this dissertation aims to contribute to the development of an automated system integrated with BLE beacon technology. The system must ensure a pleasing shopping experience for consumers while providing retailers with information to employ enhancing marketing strategies. The system must also focus on low energy consumption and reducing costs by maintaining

the system design simple. This system is intended to collect localization and proximity information of users to provide related information to their indoor position. More specifically, a smartphone application is designed to provide users with an interface and allow clients to purchase and manage product items through the smartphone application in the retail store. A web application is developed to provide the personnel of the store with a visualization tool to monitor the state of the store and its events. For localization estimation, several techniques can be employed by using the data transmitted from BLE beacons. Therefore, this dissertation presents some of the following main objectives for the development of the system:

- Development of a smartphone application to provide customers with a user interface, acting as a point of interaction between the clients and the system of the automated retail store.
- Development of a web application to provide personnel from the store with a user interface, acting as a point of interaction between the staff workers and the system of the automated retail store.
- Development of a server application of the system, assuring communication between the smartphone applications, web application and the data stored in the database. Additionally, services and all the necessary computations will be implemented to ensure all the system functionalities.
- Research and implementation of localization algorithms in the system to present personalized information to the customer based on their present position.
- Ensure communication between different devices and applications with a number of established protocols to ensure data flow over the system and allow clients to connect to the system functionalities.

1.5 Document Structure

Introduction: Chapter one presents the concept and motivation behind the system in research, what the system tries to achieve and the objectives of this dissertation to try and present an efficient new system.

State of Art: Chapter two discusses the different technologies integrated with BLE Beacon Technology, the communication protocol, the types of devices in BLE beacons, and broadcast message protocol. This chapter also presents different types of localization algorithms using beacon technology as well as the minimum requirements to integrate a BLE beacon system for an automated shopping solution. In the end, a small summary is presented to give an overview of the topics discussed in the state of the art.

System Development: In the third chapter the system development process is presented along with the methods and application techniques used. This chapter also introduces the problem the system will explore and its description to understand the expected outcome of the system architecture.

Evaluation & Testing: In the fourth chapter is presented the results of the proposed system. Using different graphs it is possible to explain the testing of different scenarios of use and the performance of the overall system.

Conclusions: The fifth chapter consists of conclusions, in this chapter is given comments about the findings in the results obtained in the evaluation in chapter four. It also stated the limitations of the system developed and explained future work. Some suggestions are addressed on the improvements or additional work that can be done to the proposed system.

2

State of Art

There are numerous approaches to developing a solution for an automated shopping system. By presenting some of the various technologies of the Internet of Things, we get to understand how we can integrate and develop a solid application. First are presented some concepts to achieve indoor localization, followed by the tools to connect the system to the clients.

2.1 Bluetooth Low Energy

Bluetooth Low Energy (BLE) was first introduced in 2010 by the Bluetooth Special Interest Group (SIG) as part of the Bluetooth 4.0 specification where was presented a new protocol architecture to enable low-power communication between devices, shown in figure 2.1. There are two main parts of the architecture, the Controller, responsible for performing radio interface tasks, and the Host, which offers higher layer functionality and support for applications [11].

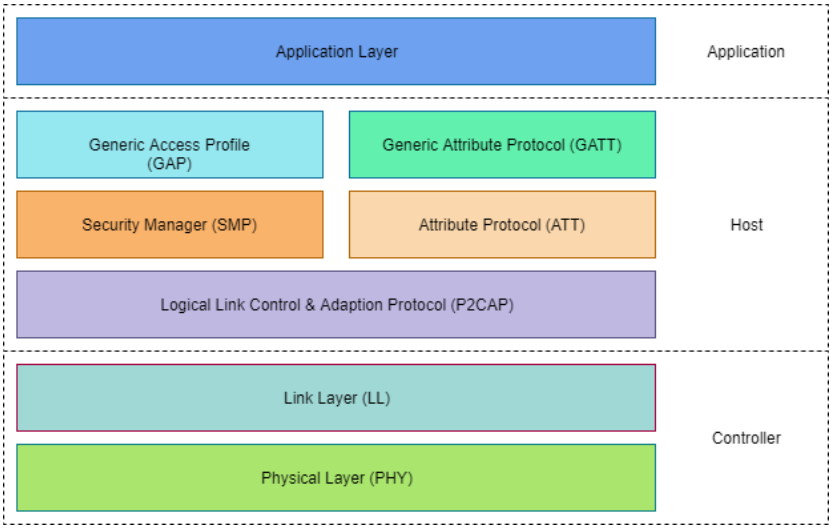


Figure 2.1: BLE Protocol Stack.

The idea behind BLE was to be able to make low-cost devices, keeping as small as efficient as possible, while supporting a range of 10 metres and operating on 2.4GHz ISM Spectrum Band, which includes the use of different standard technologies, for instance, Bluetooth classic, IEEE 802.11, etc [21]. But it was not until the release of Bluetooth 4.2 in December 2014, that Bluetooth Low Energy finally found its place in the Internet of Things, some of the features introduced are listed below. [60]

- Internet Connectivity - release of the Internet Protocol Support Profile (IPSP), compatible with Bluetooth 4.1.
- Enhanced Security - equipped with AES-CCM cryptography for message encryption and asymmetric Elliptic Curve Cryptography for key management.
- Enhanced Privacy - enhanced privacy features that consume less power than previous Bluetooth versions, adding the private address resolution in the Host and white list of private addresses at the Controller.
- Enhanced Packet Capacity and Speed - one of the most significant features, Bluetooth 4.2 have increased data throughput up to 2.5 times and capacity up to 10 times, making more efficient communication and streamlining IP connections.

With one of the latest releases by SIG, Bluetooth 5 made a significant leap forward with numerous improvements in close-range wireless communications, in terms of range, data, rate and advertising functionality doubling the data rate of Bluetooth 4.2 to 2Mbps, achievable in BLE mode, and supporting 40 meters indoor environment as opposed to 10 meters in Bluetooth 4.2, between other many enhancements specified in Table 2.1 [77]. With the introduction of BLE in the earlier versions and the latest releases like Bluetooth 5, these technologies can now be applied in a wide range of IoT systems, specifically integrated into beacon technology, which we will further discuss.

Table 2.1: Comparison between Bluetooth 5 and Bluetooth 4.2 [77]

Features	Bluetooth 5	Bluetooth 4.2
Speed	About 2 Mbps	About 1 Mbps
Range	40 meters indoor	10 meters indoor
Power Requirement	Low	High
Message Capacity	About 255 bytes	About 31 bytes
Robustness in congested environments	More	Less
Battery Life	Longer	Smaller
Security Control	Better	Less Secure
Throughput	2 Mbps	1 Mbps
Reliability	High	Low
Support for IoT	Yes	No
Bluetooth Beacon	More Efficient	Less Efficient

2.2 BLE Beacon Technology

The concept of a beacon is a device with wireless technology which broadcasts small pieces of information. While operating with BLE, beacons have a large lifespan and can run for years on a single cell of the battery [37]. BLE beacons operate by periodically broadcasting a unique identifier with data at a specific time interval. These packets when received by devices with Bluetooth technology, for instance, our smartphones, can utilize information such as the Received Signal Strength Indicator (RSSI) to be applied in localization or proximity algorithms.

With the introduction of beacon technology, many companies standardized BLE beacon protocols, Apple with iBeacon and later Google with Eddystone. These protocols were designed to implement standard advertising packet formats for messages which will be discussed in a further section 2.3 [67]. With big companies pushing new standards and introducing new hardware devices in the market for BLE beacon-based systems, beacons turned out to be more accessible to the public and to developers. As a result, there have been proposed many academic as well as industrial applications [29].

2.2.1 Localization

Global Positioning System (GPS) has proven to be a very efficient outdoor localization technology in contrast to indoor localization due to indoor conditions and occasionally on city streets as a result of attenuation and multi-path fading effects that signals suffer [29]. Solutions that use WiFi with many access points are not a feasible approach either since most WiFi location-based techniques need 3 WiFi access points at each location to provide a reasonably accurate estimation of the position [48]. So as we can imagine, to cover a region with as many as necessary access points is not very flexible and easy to deploy, to add up, power consumption is another really important factor as not only do many WiFi access points combined might cause a considerable amount of energy consumption but mobile devices are usually small and have battery power constraints [39].

Radio Frequency Identification (RFID) is another possible technology for localization but needs a dedicated reader to operate. A person with a handheld reader could read the closest tag and obtain information about his/her location but to achieve this, we need a large number of RFID tags that have to be preconfigured with location information. This method is also not very flexible and can be expensive since the costs increase with the number of used RFID readers [39].

BLE-beacon-based solutions, on the other hand, have an advantage, BLE beacons have low production costs, are easy to deploy, have very low power consumption and have easy accessibility to users. In [30] authors studied the relationship between the number of installed beacons and its positioning accuracy. The research took place in a space of 100m x 100m, where BLE beacons were deployed in a random and grid-based manner. The authors deployed 10 to 100 beacons in the random topology and placed BLE beacons with

1m to 50m intervals between them in the grid topology. In the random topology, their investigation determined that the estimated error of accuracy was lesser when the number of beacons increased, while for the grid topology, the estimated error decreased when the beacon interval increased. The feasibility of BLE beacons for localization estimation was also tested by [56] in the indoor environment, in a room with 9m width and 10.2m length. In this research were placed 3 BLE beacons, at the left, right and back walls of the room at a 1.9m height from the floor. The accuracy obtained by the authors from a single measurement and average of five measurements is studied and compared in terms of Cumulative Distribution Function (CDF). The average values of wide distance error resulting from a single measurement are 2.0 m and 1.30 m from the average five measurements, while the median values from a single measurement are 3.06m and 2.16m from the average five measurements. Thus the study concluded that the average multiple measurements can increase the accuracy of indoor positioning by 0.86m and 0.9m because it reduces the effects of multi-path fading, an effect that occurs mainly in dense indoor environments.

2.2.2 Proximity Detection

The difference between location and proximity is often misunderstood. Proximity refers to the relative distance to an object whereas location refers to the absolute position in a certain setting. BLE beacons can provide context-aware information based on the user's closeness. How proximity beacon systems work is when our smartphone, for instance, enters a particular area and catches a signal from a close-by stationary beacon, an event is triggered, allowing certain interactions between the user and the object. NFC and QR Codes are an example of technologies that could function in proximity detection systems. QR codes might interfere visually but it is a cheap technology and of ease of deployment. NFC are highly available on market devices but restricted in interacting with crowded environments since the technology has a short interaction distance, about 15 cm. Several solutions are implementing BLE beacon-based proximity detection systems since this technology proves to be very feasible in sending notifications to provide users context information. These systems work based on the Received Signal Strength (RSS) which gives a precise proximity estimation. The main limitations are that BLE beacons need to be physically deployed in the environment and also have some issues with RSS signals being affected by crowded areas, but if individually calibrating the BLE beacons, this effect can be reduced drastically [29] [3].

There have been different BLE beacon proximity systems applications in various sectors. One interesting approach was taken in [17] due to the coronavirus infection and in response to Governments' need to find digital solutions to tackle the pandemic. Here authors presented an efficient and cost-effective web-based proximity-based indoor navigation system using BLE beacon technology. In this paper, BLE beacons interacted with

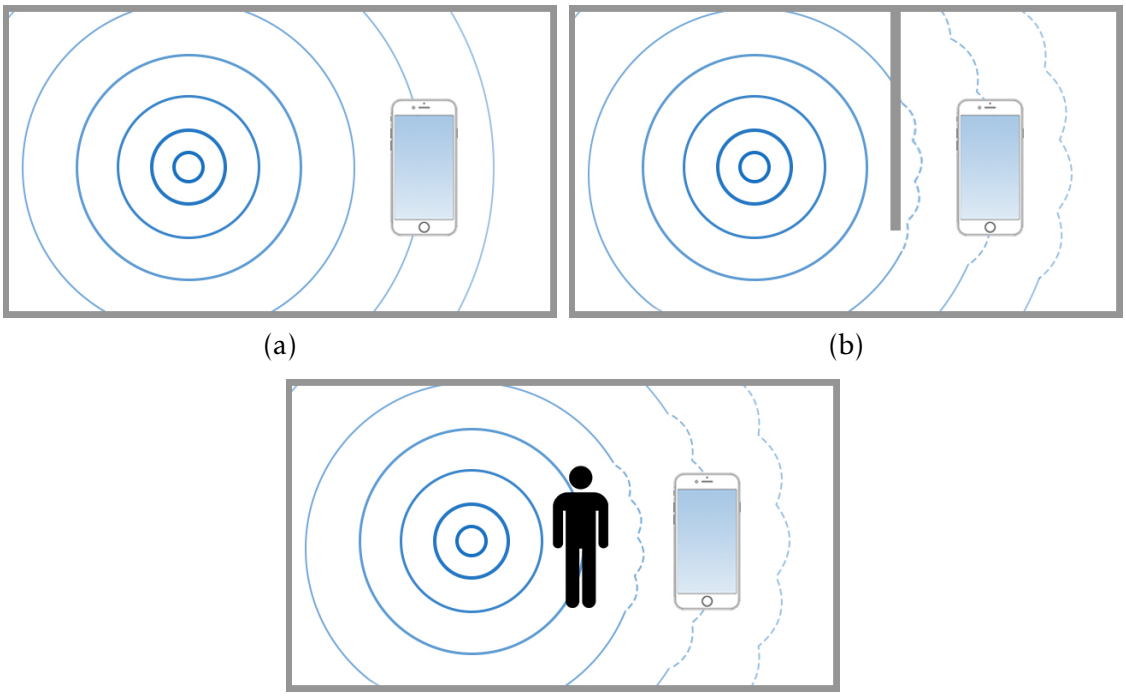
users' smartphones to perform proximity analysis, obtaining information about the environment to guide people inside the hospital based on metrics like, for instance, the lowest people density level. Another smart solution is described in [1] where the author used BLE beacons to use proximity information with the intent of automating common tasks like automatically unlocking a door when a user approaches. In this thesis, the author implemented an iOS application for Apple smartphones to perform measurements from signals emitted by BLE beacons. The author praised Apple's mobile operating system iOS for developments with BLE applications, the application's effect on the battery life was negligible, and the BLE communication latency, even though appearing to be a bit higher than expected, continued to be sufficiently small to impact the solution's performance. In [71] was designed a system with BLE beacon technology that focused on providing a better cultural experience involving context services in museums. The solution involved BLE beacons and an android Application that continuously scanned the BLE signals from beacons in the museum and chose the strongest signal providing then the user with information on the closest exhibit. The managing of the context information was treated in the application web server, very common in many BLE beacon-based applications.

2.2.3 Activity Sensing

Previously, localization and proximity services were provided by BLE beacons transmitting a user's context and location. Next will be discussed applications where the information conveyed by the beacons helps identify user activities. In [34], authors presented a BLE beacon-based solution to help care centres monitor the elderly to record the activity of multiple care receivers by estimating their present position in the care centre. To create the multi-person activity monitoring system and generate each person's activity record automatically, the system had to be able to distinguish the elderly care receivers. This operation needed to be uncomplicated since the system was dealing with care receivers who are elderly people. One of the goals was also to maintain a system that could be deployed easily with low costs. The authors decided to implement this solution with the help of BLE beacon technology. The proposed system had movable beacons and a fixed scanner. The BLE beacons were embedded into name cards that were already worn daily by the care receivers, utilizing advertisement packets transmitted by the BLE devices. BLE scanners in the area scanned the data to the Database server and with this information, it was possible to estimate the care receiver's activity and location. In [74] the main idea was to create a system that captured and identified fine-grained activities, for example transiting through a door, switch pressing and water drinking. In this application, the activities' location, movement or background context was captured by BLE beacons to give accuracy for mobile or wearable sensors to capture and trigger specific events to identify in precision the activities. Authors claim through empirical experiments of their system to be able to identify 82.2% of switch-pressing, 91.73% of water-drinking activities, and 100% accuracy in door-transitioning.

2.2.4 Limitations

As explained in the previous section 2.1, Bluetooth operates on the 2.4GHz spectrum, where it is common for signals to suffer from damping. In damping, the signal strength is reduced since they suffer from energy dissipation when, for instance, hit humans or walls causing the signal to attenuate. Without any obstacles, the signal goes directly from the beacon to the device, which provides the highest accuracy, but the accuracy can be reduced if there is a wall in between the device and the beacon. Human bodies are 80% consisted of water thus also affecting the signals' propagation shown in Figure 2.2 [33]. The approach to take here is to place beacons at a high level of height if the environment where the beacons are set in has a lot of obstruction from objects and walls or if the region where beacons will be operating need to work with a crowded area of people.



(a) (b)

(c)

Figure 2.2: Propagation of signals in different environments.

2.3 Beacon Protocols

BLE beacons periodically broadcast specific types of messages. What makes BLE beacons so attractive is the fact that devices don't need to be paired to receive the broadcasted signals. Several beacon protocols were introduced by companies like Apple, Google and Radius Networks to provide BLE beacon technology with a standard format for these types of messages. In the sections below will be presented the most popular message beacon profiles.

2.3.1 iBeacon

iBeacon was introduced by Apple at the World Wide Developer Conference (WWDC) in 2013 and uses BLE which is part of the Bluetooth 4.0 technology. In the case of using a smartphone, the operating system needs to support the iBeacon profile, which nowadays is compatible with both Apple and Android smartphones since Apple integrated the iBeacon support in iOS version 7.0 and Google integrated with Android version 4.3. BLE beacons do not receive requests from other devices, there is only one-way communication, BLE beacons broadcast their messages to other devices. In the iBeacon communication protocol, the data format has a 25-byte payload with a unique ID. This frame consists of a UUID, a Major, a Minor and the Measured Power, presented in Figure 2.3 [33].

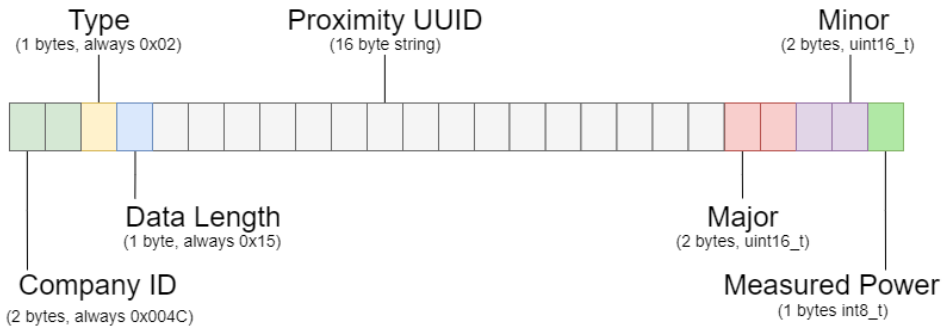


Figure 2.3: iBeacon Data Format.

UUID: This is a 16 byte string used to differentiate the device between a large group of related beacons, i.e "fb0b57a2-8228-44 cd-913a-94a122ba1206".

Major: This is a 2 byte unsigned integer used to distinguish a smaller subset of beacons within the larger group.

Minor: This is a 2 byte unsigned integer used to identify individual beacons.

iBeacon can work in two functions, one of them is creating a region monitoring around a smartphone, which when entering or leaving a beacon's region of operation, a notification is pushed or the app can run a specific programmed function. One of the main advantages of iOS operating systems is that the app does not have to be running, therefore if the user has the Bluetooth and location service turned on, the iOS can detect the iBeacon data sent by the beacons and execute the corresponding code for the event instantly. Another iBeacon function is the ranging function. This function calculates the proximity between the beacons and the device. The iOS API presents four states specified in the iBeacon protocol, Immediate, Near, Far and Unknown. The 'Unknown' state means that no distance could be calculated to the beacon or that the user is more than 30 meters away from the beacon, the 'Far' state is between 2 and 30 meters, the 'Near' state is between 2 and 0.5 meters and the last state 'Immediate' is within 0.5 meters (Figure 2.4) [33].

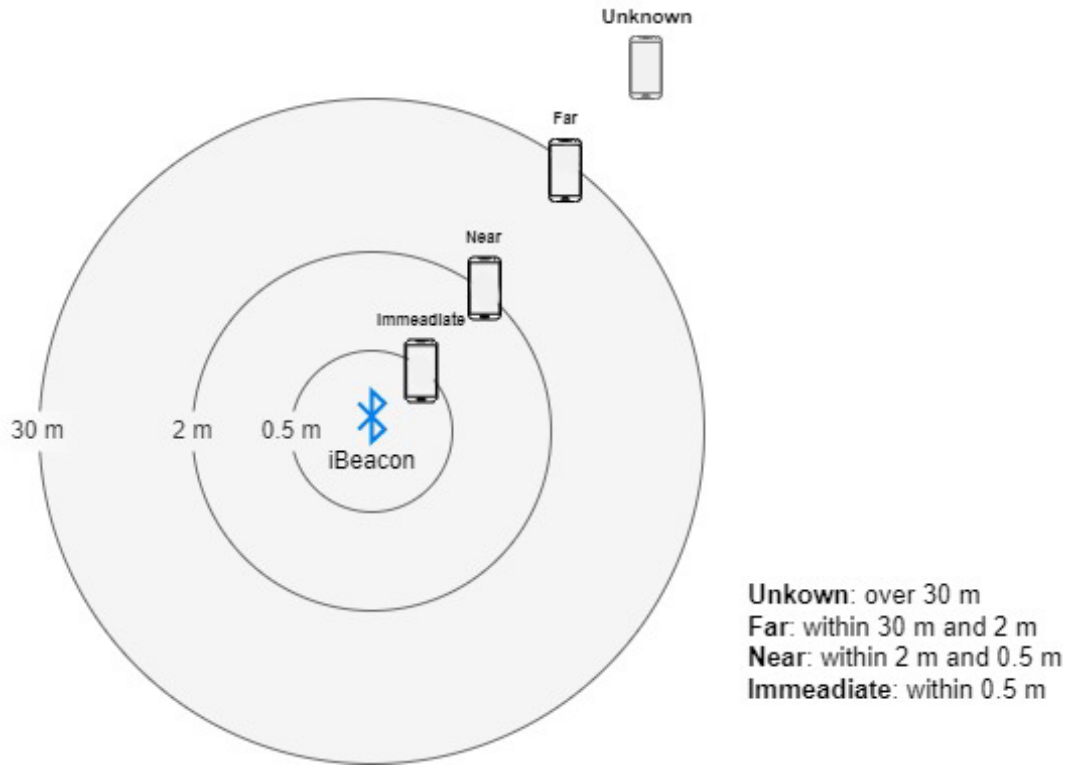


Figure 2.4: iBeacon Distance States.

The RSSI is used not only to determine the proximity to a particular beacon but also to calculate the accuracy of the proximity estimation. The higher the signal strength, the better the proximity measurement. Developers can also get the RSSI values from the iOS API to apply different algorithms from which the API uses, for example, giving developers full freedom to work on new implementations.

There are some limitations considering the iBeacon protocol. If developers want to use iBeacon functions in iOS they have to stick to the data format provided by Apple. Another issue is that an iOS smartphone, while using the iBeacon protocol, can detect beacons even if the application is closed, whereas, Android operating systems do not support this feature [33].

2.3.2 Eddystone

To compete with Apple's iBeacon standard protocol, Google launched their open-source protocol specification that defines BLE message format for proximity beacon messages called Eddystone. Differently from the iBeacon protocol, Eddystone allows interaction with the Chrome browser which comes installed almost on any operating system. Eddystone provides flexibility in contextual content development since it requires no mobile application on smartphones to interact with a deployed beacon. The protocol describes different frame types, Eddystone-UID, Eddystone-URL and Eddystone-TLM shown in Table 2.2, which can be used individually or in combinations depending on the type of

the application. [68]

Table 2.2: Eddystone Data Format.

Eddystone Data Format						
UID	1 byte	1 byte	16 bytes		2 bytes	
	Frame Type	Ranging	UID		Reserve	
URL	1 byte	1 byte	18 bytes			
	Frame Type	Ranging	URL			
TLM	1 byte	1 byte	2 bytes	2 bytes	4 bytes	4 bytes
	Frame Type	TLM Version	Battery Level	Temperature	ADV_CNT	SEC_CNT

UID: This is a unique 16 byte Beacon ID composed of a 10-byte namespace ID and a 6-byte instance ID. The Beacon ID is used to map a device in external storage, whereas the namespace ID is used to group a particular set of beacons and the instance ID identifies the individual devices in the group.

URL: The URL is compressed and encoded in a frame and once it is decoded, the URL can be used by any client with access to the Internet. The operating principle is identical to the conventional QR code.

TLM: This is a frame that broadcasts telemetry information about the beacon itself, in particular the value of the battery voltage, device temperature, number of broadcasted packets, number of seconds that the beacon has been powered and the packet version.

Eddystone is supported by both, Android and iOS devices. The working principle of the Eddystone protocol is very similar to the iBeacon, the beacons broadcast an ID which is then used by an application on the smartphone to get data from the database. With the URL, Eddystone gives an optional working approach since the URL can redirect the user to the corresponding web page and the URL can be mapped with the Beacon's ID in the database. In 2021, Google shut down their cloud-based system for beacon management and Google Play Services, but Google's Eddystone Bluetooth beacon standard is not deprecated. People often confuse the abandoned system with the beacon standard, but to not forget that Eddystone protocol is open source and that Android Beacon Library still supports Eddystone. Therefore, Eddystone is a great beacon standard protocol with many advantages, in the public domain and is published on Google's GitHub account [13] [68].

2.4 Localization algorithms

Indoor localization is the key technology for location-based services, which have a critical impact on a broad number of industries, especially in the retail industry. In outdoor localization, GPS can provide an error span of up to 5 meters but indoor localization demands high precision. Due to multi-path propagation in indoor environments, accurate indoor localization is extremely difficult to achieve and owing to a considerable need

for accuracy in different applications there is far less room for error. With the rapid development of wireless technology, multiple types of new communication technologies were introduced such as WiFi, Zigbee, Long Range (LoRa), Radio Frequency Identification (RFID) and BLE. With the rising popularity of BLE and smartphones with BLE functions, Bluetooth-based indoor localization has increasingly attracted its use in system solutions. BLE-based positioning systems locate mobile devices using the RSSI value from BLE beacons in an indoor context.

There are several algorithms used in indoor positioning. Most complex algorithms are based on arrays of antennas at the transmitter and receiver, on the angle of arrival, time of flight, etc. As a result, these types of algorithms are not convenient for indoor localization. Commonly, applications in indoor localization tend to use the RSSI to determine the position of the signal receiver device. The next sections will be presented the most commonly used algorithms, filtering algorithms for preprocessing the RSSI values and different types of algorithms for determining localization like the lateration or fingerprinting algorithms [51].

2.4.1 Filtering Algorithm

As mentioned before, RSSI is the most common measurement used in localization algorithms. The issue with the RSSI is that signals are prone to suffer from multi-path effects, whether being refracted, diffracted or blocked by obstacles in indoor environments. These effects cause RSSI to fluctuate in value over time. However, there are ways to reduce the error that may occur with the RSSI values through the use of filtering algorithms. Kalman and Moving Average filters are two of the most popular filters used to correct these errors in a preprocessing phase [65].

2.4.1.1 Moving Average

Moving average is very simple to implement with the RSSI. In the Moving Average filter, N samples of RSSI values are gathered and averaged to create a new one, usually obtaining a far better location result than by only using a single RSSI value. The algorithm formula is displayed in 2.1. [65]

$$\overline{RSSI} = \frac{1}{n}\sum_{i=0}^{n} RSSI_i \tag{2.1}$$

2.4.1.2 Kalman

The Kalman filter is very popular for its light computational requirement. The filter has two stages, the prediction stage and the update stage. How the filter works is, in the prediction stage, the Kalman filter receives an RSSI value, compares it to the previous values obtained and estimates a new value and the error in between them. In the update

stage, various variables are updated for the next calculation [65]. The formulas are as follows:

Prediction Stage

Predict the current state:

$$x_k = A * x_{k-1} \tag{2.2}$$

Predict the error covariance:

$$P_k = A * P_{k-1} * A^T + Q \tag{2.3}$$

Update Stage

Calculate the Kalman Gain:

$$K = P_k * H^T * (H * P_k * H^T + R)^{-1} \tag{2.4}$$

Calculate new state:

$$x_k = x_{k-1} + K * (Z_k - H^t * x_{k-1}) \tag{2.5}$$

Calculate the new error covariance:

$$P_k = P_{k-1} - (K_k * H * P_k) \tag{2.6}$$

Z_k: Measurement vector for current time. x_k: Estimate of the current state. P_k: Estimate of average error for the current state. A: State transition matrix. K: Kalman gain. H: Observation matrix. Q: Estimated process error covariance. R: Estimated measurement error covariance.

In [65], authors using BLE beacon technology, determined the optimal density of beacons that could be placed in an environment to improve localization accuracy. In this experiment, the authors compared the performance of two localization techniques using the RSSI signals with the help of two types of filters, such as the Moving Average and Kalman filters. Through experimental analysis, authors stated that both filters provided similar increases in performance, Kalman filter presenting a slightly higher average accuracy and the Moving Average filter was able to provide higher average precision. Implementing average filtering improved accuracy by 24.49% whilst Kalman filtering improved by 27.05%, compared to the localization algorithm without filtering.

In [38], authors developed a simple BLE beacon-based indoor navigation model in three different topologies to prove the practicality of these types of systems. For the navigation, the authors used a Google Nexus smartphone as a receiving device. To increase the accuracy of the system, a Kalman filter was developed to correct noise caused in the

environment. One of the most interesting topologies was a 6 x 6 m topology which contained different obstacles like desks and chairs to eliminate the line of sight between the beacon and the smartphone. As expected, the RSSI values provided poor accuracy due to the multipath fading effect in the environment. Using the Kalman filter improved the system performance by 34.2%. One additional interesting aspect of this research was that the filter was applied in the developed Android application without the need for any back-end computing. This shows the already mentioned light computational power of the Kalman filter. Undoubtedly, the Kalman filter seems like a great addition to a system for increasing the accuracy of location-based services.

2.4.2 Distance Estimation

To perform the several existing location algorithms it is necessary to calculate the distance between a target and a reference point. As described in previous sections, BLE beacons make use of RSSI values to provide the location of a device. As a result of the free-space path loss phenomenon (FSPL), the distance between the transmitter of the signal and the receiver can be expressed by the Friis transmission formula.

$$FSPL = (\frac{4\pi d}{\lambda})^2 = (\frac{4\pi df}{c})^2 \tag{2.7}$$

The d represents the measured distance in meters between a receiver and a transmitter, λ corresponds to the wavelength in meters, which can be replaced with the quotient between c (speed of light) and f (frequency). Considering the inverse square law which establishes a relationship between the signal strength and distance, the equation is squared. Due to multipath effects, this relationship may take different values, so we exchange the power of 2 by the propagation constant γ, presenting the intensity loss of a signal to the environment in propagation.

$$FSPL = (\frac{4\pi df}{c})^{\gamma} \tag{2.8}$$

We can then express FSPL in dB and separate the equation 2.8 into a sum of two factors.

$$FSPL(dB) = 10log_{10}((\frac{4\pi df}{c})^{\gamma}) \tag{2.9}$$

$$FSPL(dB) = 10\gamma log_{10}(d) + 10\gamma log_{10}(\frac{4\pi f}{c}) \tag{2.10}$$

The second part of the sum in equation 2.11 can be replaced by a constant called Measured Power, usually denoted with A. This constant is given by the manufacturer of the device (beacon) and concisely means the value of the received signal strength at a distance of 1

meter. The distance can be finally calculated using the signal strength since each beacon transmits the value of constant A, a method called RSSI.

$$RSSI = -10\gamma log_{10}(d) + A \tag{2.11}$$

The value of the propagation constant depends on each environment, depending on the number of obstacles or multipath effects in general. The value of this constant needs to be estimated for the particular environment if we want to achieve the highest level of accuracy [4].

2.4.3 Lateration Algorithm

In the Lateration Algorithm, the distance information between the target and multiple reference points is used to calculate the location of the target. The distance between the target and the reference point can be calculated by measuring the RSS values or time delay and converting those values to distance. Time of Arrival (ToA) systems measure the time a signal takes to travel from a reference point to a target. To implement these types of systems, three reference points are needed and by measuring the three different distances, the location of the target can be estimated. Synchronization between the target and the reference points is essential since a very small difference in clock offsets will result in accuracy errors. ToA algorithms types are only used in ultra-wideband systems since the signals operate in broadband, thus not suffering from multipath effects in indoor environments. The RSS in lateration algorithms is used to measure the signal strength emitted at the target from the reference points. These signals are converted into distance values to obtain the location of the target. But as discussed before, the major problem with this type of signal is the multipath fading effects of the environments. RSS algorithms are commonly used in BLE beacon systems to locate the targets. With the support of filtering algorithms, the errors caused by the environment can be highly reduced as described in 2.4.1 [10]. The most commonly used lateration algorithm in BLE beacon-based systems is the Trialteration algorithm.

2.4.3.1 Trilateration

The Trilateration algorithm is used to compute the 2D position of a target. To achieve this, three reference nodes with known locations are selected. If the target detects more than three nodes, the nodes with the highest RSSI values will be preferred since it indicates that these nodes are the closest to the receiver. Given the distance between the target and the reference nodes, calculated using the path-loss model, three circles are drawn, as shown in Figure 2.5. Trilateration then calculates the intersecting point of the three circles and the result will give the exact position where the target is located [69]. The intersecting point is calculated using the three-circle equations which is equivalent to

solving a set of three equations with two unknown variables shown below.

$$r_a^2 = (x - x_a)^2 + (y - y_a)^2 \tag{2.12}$$

$$r_b^2 = (x - x_b)^2 + (y - y_b)^2 \tag{2.13}$$

$$r_c^2 = (x - x_c)^2 + (y - y_c)^2 \tag{2.14}$$

The x and y variables represent the coordinates of a point that corresponds to the position of the target that we are determining. The measured distances between the reference nodes and the target are presented as r_a, r_b and r_c. The rest of the variables are the known coordinates from the reference nodes' location. The reference points are defined according to the local coordinate system of the area where the beacons are implemented for instance. The distances applied in trilateration must be first scaled to the same coordinate system as the reference points [4].

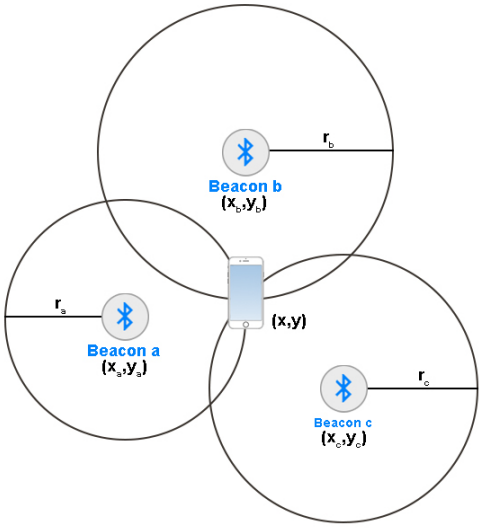


Figure 2.5: Trilateration with 3 beacons and a smartphone.

In [4], authors implemented a BLE beacon system for indoor positioning. The user's position was calculated using the trilateration algorithm with the RSSI from the beacons. The study conducted both static and dynamic testing to measure the performance of the method. The authors stated that the implemented system while using some preprocessing to eliminate multipath effects errors, satisfied the position accuracy obtained with this algorithm in static analysis with an average error of 1.75 meters. In 90% of the measurements, the accuracy varied from 0.11 meters to 2.53 meters. In the dynamic analysis, the authors observed a drop in accuracy in some traversed paths because as stated in the paper, these areas received weak and unstable signals to be able to determine the positions. The authors recommended an increase in the number of beacons used in a room as well

as a decrease in the distance between the beacons, stating that it significantly improved the positioning accuracy in the tested dynamic scenarios.

In [24] is presented an indoor location-based system to locate the indoor position of a user for a particular service. This system is composed of a few elements, namely the service-provision client, user application positioning technology and the localization server which is responsible for processing the information coming from the user application to localize the user's device. In this system, Bluetooth, RSSI and the trilateration algorithm are used for the position localization technology. The position information is obtained from the BLE beacons technology which transmits the data to the user application. The authors tested the system in a room consisting of 7.2 meters and 4 metres in length and width. The performance evaluation showed through the distribution graph for the experimental estimated location that the accuracy in the conditions described in the study resulted in approximately 74%. This result was obtained in a setting where no obstacles were in the testing space. With many obstacles the accuracy lowered thus authors increased the accuracy by setting a shorter update cycle in the indoor system's map.

2.4.4 Centroid Algorithm

Centroid localization is largely employed in applications due to its simplicity and robustness to path loss effects. Taking into account that the path loss differs in each environment, a centroid localization algorithm does not model the beacon's energy distribution. Taking BLE beacons devices as an example, the principal concept of centroid localization is that when a device is within the range of a beacon, inside its communication region, an imaginary polygon is created between each detected beacon as shown in figure 2.6 and the centroid is determined. To estimate the position (x, y) of a device that detects n beacons, the arithmetic mean of the beacon positions is calculated [32].

$$x = \sum_{i=1}^{n} x_i w_i, \quad y = \sum_{i=1}^{n} y_i w_i \tag{2.15}$$

with x_i and y_i representing the position of the ith detected beacon and its weight wi. The weight reflects the impact of a particular beacon, increasing the weight of a beacon will result in a greater contribution from that particular beacon in the location estimation. The traditional Centroid Localization (CL) weights all beacons equally, meaning that each beacon will have the same contribution, even if the beacon transmits a low signal or is the furthest beacon from the device [32].

$$w_i(CL) = \frac{1}{n} \tag{2.16}$$

The Weighted Centroid Localization (WCL) weights the position of beacon i by the inverse of its distance di to the device in proportion to the sum of inverses of the n beacons'

distances. These values can be calculated by distance estimation described in section 2.4.2. An additional attenuation factor g is included.

$$w_i(CL) = \frac{\frac{1}{d_i}^g}{\sum_{j=1}^{n} \frac{1}{d_j}^g} \tag{2.17}$$

In the Compensated Weighted Centroid Localization (WCWCL), the RSSI pi value is presented as a logarithmic decibel value. Thus, when assigning WCWCL weights, if a beacon has a weak signal, the beacon will have less influence on the calculated position than for instance with the WCL algorithm. The only calculations needed prior to calculating the location of the devices is determining the beacon positions as well as the WCL attenuation factor g. The more signals the device target receives from beacons, the higher the accuracy in the calculation of the target's localization [32].

$$w_i(WCWCL) = \frac{10^{\frac{p_i}{10}}}{\sum_{j=1}^{n} 10^{\frac{p_j}{10}}} \tag{2.18}$$

In [70], authors studied the performance of the WCL using the RSSI from BLE beacons. This study reviewed the WCL algorithm in their testbed building and analyzed the configuration parameters of the algorithm for indoor positioning. The WCL was tested by using BLE beacons for transmitting periodically advertisement messages and an iOS device as the receiver of these messages. The authors developed an iOS application to display the estimated location by the algorithm. The measurements took place in two different regions of a corridor, at the centre and the border, the four closest beacons were selected. The authors claim that the error measurement of the location was slightly higher in the border regions than in the central regions. The authors also proved that the best value performance of the attenuation factor of the WCL algorithm was 0.5 in the WCL weight calculation of the beacons. The lowest and the highest average errors obtained in each of the regions were 1.58 and 2.45 meters.

In [32] is presented an evaluation of the performance of BLE indoor positioning algorithms, namely, trilateration, fingerprinting and centroid. The study was taken place in an in-house fair where, with the use of BLE beacons and smartphones, the location of visitors was calculated among exhibition stands. The authors studied how the localization algorithms responded to the impact of the crowd. Data were collected before the event in the same location but uncrowded to investigate how the positioning algorithms coped with unstable RSSIs when the environment turned too crowded. In the study, during the uncrowded stage, the authors set up eight beacons on the floor in a 15 by 15-meter space of the event where the only obstacle in the environment was the researcher roaming the open space, no other obstructions were in the space affecting the signal propagation. On the crowded stage, exhibition booths and visitors were populating the in-house space fairly. The authors could not set up the beacons on the ceiling due to structural limitations, therefore the beacons were attached to the exhibition stands. The problem with this

is that beacons on the ceiling allow frequent line-of-sight connections with smartphones. The fair space had an area of 20 by 40 meters and 27 beacons, more beacons than in the uncrowded stage but the authors claim that this setup did not increase accuracy. The study concluded that the weighted WCWCL achieved the highest position accuracy and outperformed the other weighing methods like CL and WCL in the crowded space with a positioning error at the 95th percentile of 5.6 meters and a difference of 1.4 meters to the uncrowded setup.

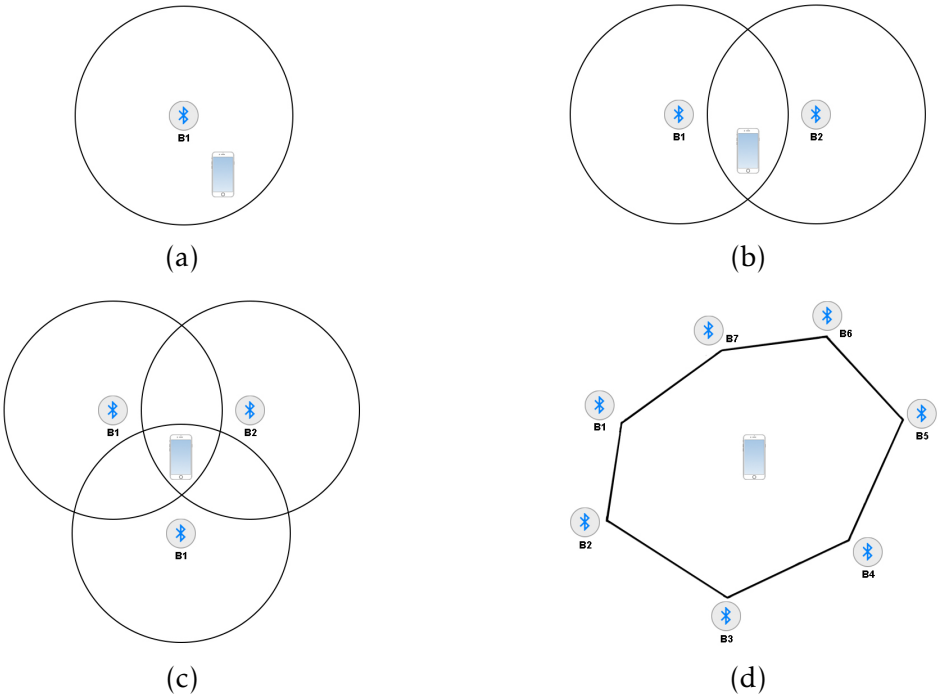


Figure 2.6: Centroid imaginary polygon.

2.4.5 Fingerprinting Algorithm

In the Fingerprinting algorithm, there are usually two stages, training and positioning. In the training, stage is constructed a fingerprint database whereas in the positioning stage is estimated the location of targets is by comparison of the current measured signal strength with the ones in the database. The database is constructed by taking reference points and measuring the signal strength received in these precise locations. By doing this, the characteristics of each of the reference points are obtained and mapped according to each location in the database. In the positioning stage, by measuring the signal strength received by a target, the result is then compared with the data mapped in the database with a matching algorithm. The output of the positioning stage is the estimation of the target's location. It is also common in fingerprinting to use localization algorithms in the training stage when building the mapping of the reference points in the database as well as using the same algorithm in the positioning stage and comparing with the data of the

database.

Commonly, for the matching algorithm, a deterministic or probabilistic method is used. In the deterministic techniques, having the signal strengths measurements mapped with the reference points in a database, methods like the Euclidian distance or Minkowsky distance can be applied to determine the distance between the current fingerprint and the offline fingerprints. The smallest distance is then used to estimate the user's coordinates. The K-nearest neighbour method can also be used, reducing the computational complexity by only taking into consideration the nearest K fingerprints from the database.

The performance of fingerprinting techniques increases when a large number of reference points is measured. But numerous fingerprinting measurements in the training stage is very time consuming and not convenient when the indoor environment is large or the setting in the environment constantly changes, like for instance in a retail shop. [10]

In [16], authors studied the unique proprieties of BLE signals and the application of fingerprinting to locate BLE devices in an environment with BLE beacons. The study evaluates beacon parameters and their impact on positioning performance. The testbed used for the study was a 600 m^2 office in normal daily use. For BLE beacon positioning, a Bayesian estimator and fingerprinting were used. There were 19 beacons deployed, installed on top of windows, ledges, walls, most of them at a height of approximately 1 m from the floor and oriented to provide the best response in the horizontal plane. Authors claim to have achieved tracking accuracy of < 2.6 meters 95% of the time while using a dense beacon distribution, 1 beacon per 30 m^2, and < 4.8 meters when using a lower density distribution, 1 beacon per 100 m^2. This result was achieved in a rarely crowded space, authors suggest that these results would be different in a crowded setting and most challenging.

In [52] is presented a comparative analysis between different types of positioning algorithms to investigate which of the algorithms showed the highest accuracy. In this study, the authors implemented several beacons in an environment and used a smartphone as the target for location estimation. The position algorithms compared are Proximity, Centroid, Weighted Centroid, Trilateration and Fingerprinting. The testing was conducted in different environments, one of them a 4.64 by 4.64 meters room. In the room, 4 beacons were set up on each wall and at the same horizontal level. According to the authors, from the experimental results, the Fingerprinting algorithm had a distinguishable higher accuracy than the other algorithms tested with a 0,65 meters error range, being the Weighted Centroid the second-best performed algorithm with a 0,97-meter error range.

2.5 Retail System

Previous sections covered several technologies and various localization techniques with distinct advantages needed to achieve localization while using BLE Beacon Technology. This section presents common applications implemented in systems with beacon technology integrated. The applications presented can be used in the development of the

automated retail shop system with all the previously addressed technologies. The beacon system in a retail shop needs to cover several functionalities, from processing beacon signals to locating the device, displaying the existing products near users, providing user management, processing payments, etc.

Beacons have only one-way communication since they can only transmit information, hence they aren't capable of processing requests coming from other devices. To process the data transmitted from beacons, a device is needed with Bluetooth technology. In previous sections, many researchers used smartphones to process the beacon signals. Smartphones are small, handheld devices that have been integrated with numerous technologies throughout the years, Bluetooth being one of them. With the extraordinarily rising rate of smartphone usage in the world [57], people nowadays travel everywhere with their smartphones. Therefore, smartphones seem like a decent selection device to be used in beacon localization techniques while being able to also serve a user interface to the clients of the store. In the commented research from previous sections, the authors had to implement a smartphone application to process the signals transmitted by beacons.

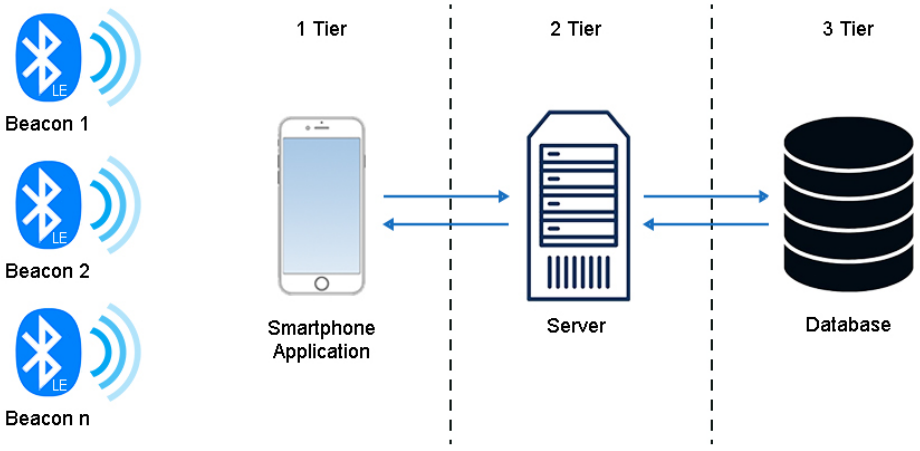


Figure 2.7: Common Beacon System Architecture - Three Tier Architecture

To address the complexity of retail systems, researchers implemented the popular client-server architecture that is displayed on Figure 3.2. The use of a distributed system with a client-server architecture seems like a prominent application for splitting the complexity of the system into different layers. A distributed system is a set of hardware components and software applications for implementing functions like processing, storage, transmission and data protection [35]. The Client-Server architecture is designed to partition functionalities and resources between the service providers (servers) and service requesters (clients). The three-tier architecture commonly used in beacon systems is shown in Figure 3.2. The goal of this architecture is to provide flexibility, interoperability,

scalability and usability while hiding the complexity of distributed processes from the clients. The architecture is composed of, a presentation tier, where a smartphone application (SA) is integrated, an application tier, providing process management services in the backend/server, and the data tier which is where the information processed by the system is managed and stored. The main idea of using this type of architecture is to separate the interface logic, processing logic and data logic, making it a suitable architecture to be adopted in the beacon system of the retail store [59] [55].

2.5.1 Smartphone Application

The presentation tier is one of the most important parts of the architecture of the beacon system to be implemented in the retail store, as mentioned, this tier is composed of a smartphone application. The appealing use of apps for consumers is due to their ability to store referenced information, save critical data, the ability to perform difficult calculations and the easy access to the internet, all through a user interface [19]. One of the main functions of the smartphone application is to provide clients with a user interface in the retail beacon system environment. The purpose of designing an interface for the mobile application is to increase customers' interest and productivity as well as to act as a shopping middle-ware between the system and the client. The smartphone application interface can showcase the product items relative to the client's localization, provide cart management inside a store, process payments, amongst other features. Another main function of the smartphone application is to interact with BLE beacons, being responsible for measuring RSS in the background by discovering beacon signals, potentially even estimating distance or applying the filtering algorithms on the received signals. The

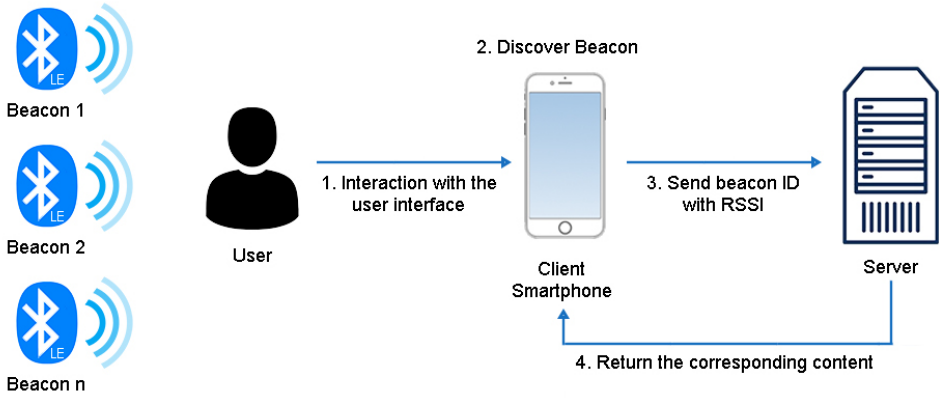


Figure 2.8: User interaction with a beacon system.

information can then be sent with a unique beacon identifier to the second tier of the architecture, the server, through an HTTP request to get related information. To interact with the server and the database for sending and getting data, clients can use the web

services provided by the server which will be described in the next section. An example of the interaction between the user and server in a beacon system is exemplified in Figure 2.8. The network requests used to interact with the second tier can be POST, PUT and GET methods. The POST method can be used for updating the content on the server side, for instance, a product added to the cart of the client. The GET method can be used for getting information from the server like user information. The PUT request can be used to create resources in the server for example the RSS values received by the smartphone application. The smartphone application can receive the product information based on the user's localization, personal advertising information to be shown in the user interface, etc. As the authors state in [7], a simple interface design, information hierarchy and visual display of attributes contribute to a very positive evaluation when users interact with their smartphones.

2.5.2 Server

The second tier of the architecture is the server, responsible for providing communication with multiple clients (smartphones) and the database in the beacon system of the retail store. Considering that n beacons interact with one client at the same time, there will be n requests sent by the client to the server with the signal information. Since the system might have m clients interacting with the server, there will be a total of mxn requests being sent to the server with several responses sent back. With a scaled beacon network, more clients will be interacting with more beacons and the requests will increase proportionally, hence the importance of having a server tier [29]. The server's main functionalities are sharing data, resources and performing computation. The server can be responsible for performing the localization algorithms with the RSS values measured and sent by the clients. When receiving the result of the localization calculation, the server can access the database to return the related product items in that location. The server can also access the database to send clients with advertising information, user data, etc. The communication between the server and the clients can be provided by the server's web services. The web services use REST-style architecture as shown in Figure 2.9.

Representational State Transfer (REST) was first introduced by Roy Fielding, who described a set of constraints for the architecture where systems are characterized as stateless and separate the concerns of the client from the server. REST is used in a client-server architecture where requests are made from clients to servers, when processed, the servers return a response. These requests and responses are built around the transfer of representations of resources. A resource is an object which is identified by a Uniform Resource Identifier (URI), while a representation of the resource is normally a document that captures the current or intended state of a resource. The REST language uses nouns and verbs. REST introduced several design principles, addressability, statelessness and uniform interface. In addressability, the datasets are modelled to operate as resources and to be accessed using a specific URI. To access the REST resources, a uniform standard

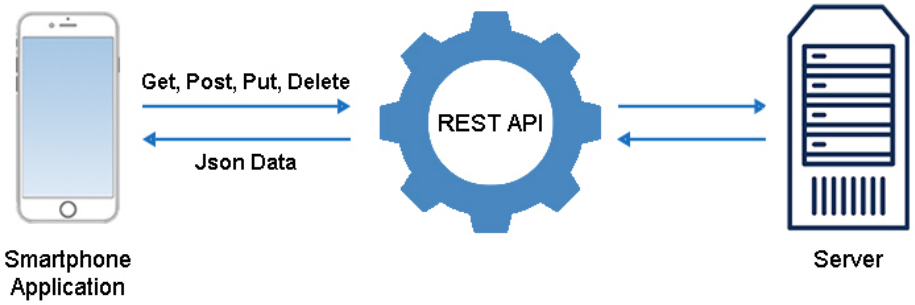


Figure 2.9: Rest API Model.

interface is used, for instance by using HTTP methods. Using this REST interface, several clients can hit the same REST endpoints, even performing the same actions and receiving the same responses. Every transaction is independent since data is unique from a specific request and since the client session data is not maintained on the server (statelessness), the server responses are also independent. Any system which uses REST architecture is using a RESTful web service. RESTful web services use HTTP methods like GET, PUT, POST and DELETE to retrieve, create, update and delete resources on the server. The response from servers can come with a payload formatted in JSON or another format, also a confirmation of the resource state change can be sent [47].

2.5.3 Database

The third tier of the beacon system architecture is the database, responsible for storing the collection of data of the system. There are two main model types of databases, the relational and NoSQL databases. Developers must choose the most suitable type of database to use in their system based on the features provided by each type. Relational Databases were introduced over 30 years ago to provide business data processing to deal with information storage from financial records, personal data, etc. NoSQL databases made their first appearance in 2000 when highly interactive applications like social media and e-commerce had to process and store huge amounts of data. For Web 2.0 and the recent web 3.0, interactive features in databases are highly sought to support their needs. Since relational databases failed to match these requirements, NoSQL databases made their replacement by proving to be efficient in dealing with such demands. However, Relational Databases still have a strong place in different applications, mostly deployed in data systems that require high levels of consistency.

The relational Databases follow the fundamentals of the ACID model, Atomicity, Consistency, Isolation and Durability. With Atomicity, the integrity of transactions is guaranteed, Consistency, the data stability is provided in a database, Isolation, each

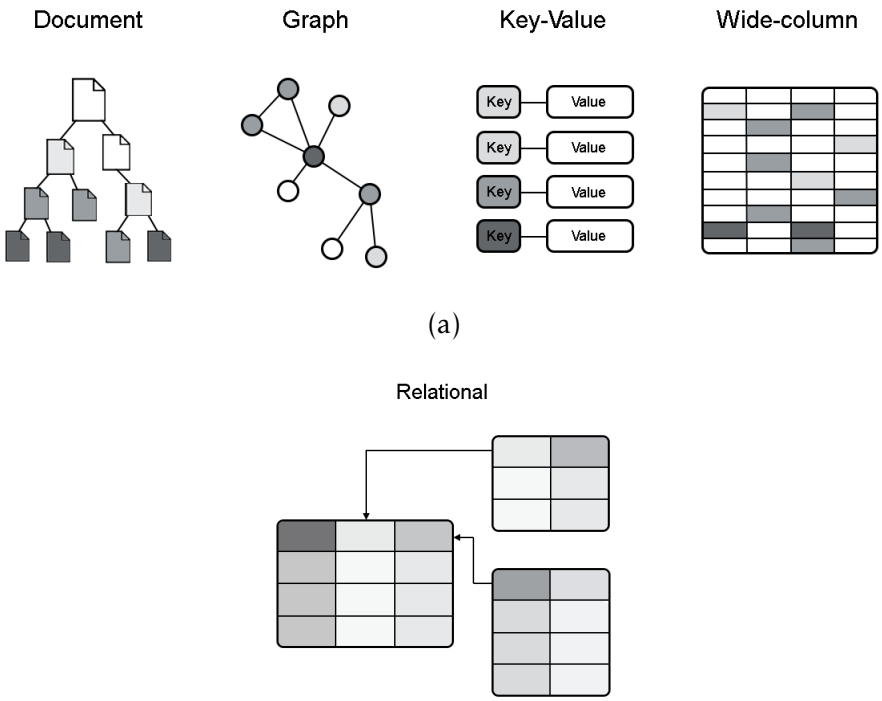


Figure 2.10: NoSQL and Relational types of databases.

transaction is independent and there can be several transactions executed in the database at the same time, and for last the Durability which ensures that stored transactions do not change their state even if there occurs a failure. Relational Database's popularity is due to ACID's model properties, being consistency and availability the main ones. In contrast, NoSQL Databases follow the BASE model, Basically Available, Soft State and Eventually Consistent. Available means that some Data from NoSQL Databases can be only partially available when some of the distributed database's parts are not operating or cannot be reached. Soft State means that the data can be modified in time even without input. At last, Eventually Consistent means that data may only become consistent shortly and not right after operation execution. By using the BASE model, NoSQL is provided with the ability to be scalable and offer users great levels of availability. There are many types of NoSQL databases but the main types are, Key-Value (e.g. Memcached) , Document Oriented (e.g. MongoDB), Column Databases (e.g. Cassandra) and Graph Databases (e.g. GraphDB), shown in Figure 2.10. There are no actual better types of Database, Relational and NoSQL databases will exist alongside each other, it depends on the needs of the application. While Relational Databases lack the support for unstructured data, this type has better consistency, security and offers a standard query language. This type of database can be used to store data with relational needs like user information, product information, etc. On the other hand, NoSQL lacks standardization and has a lack of

security, but this type of database provides great scalability and flexible schema, also offering cheap and better performance. This type of database can be used to store data of the RSS signals coming from the beacons from each client as described in the previous sections about the number of requests that are sent from clients. Ideally, both types of databases can be used in the same system to support different types of data [36] [49].

2.6 Summary

In summary, this chapter demonstrated the distinct existing technologies and methods to be integrated with an indoor automated BLE beacon-based system. The introduction of Bluetooth 4.0 technology-enabled low power communication between devices and the development of what are called, Bluetooth Low Energy devices. With technological advances in the present day, this communication technology has faced numerous improvements in speed, range, power requirement, etc, which lead to the development of BLE beacon technology, small low-power devices with wireless technology. Their practicability and serviceability provided great use in different applications, for instance, Localization, Proximity Detection and Activity Sensing. Also, the development of different communication protocols, designed to standardize the format of beacon message profiles, enabled different formats of establishing communication between BLE beacons and smartphones which paved the way for innovative applications in BLE beacon systems. However, BLE beacon technology has some limitations to its efficiency and feasibility as BLE signals operate on a 2.4GHz spectrum band, where signals tend to suffer from multipath effects in obstructive environments. Therefore, many projects employed filtering algorithms to minimize the errors created by these indoor effects. By using these preprocessing algorithms and with the current existing localization methods, accurate localization estimation of devices in indoor environments can be achieved, even with high obstruction. By implementing a 3 tier client-server architecture, composed of clients, a server and, a NoSQL or Relational database, we can produce a simple but efficient distributed system with high flexibility and interoperability. Integrating web services in our system can effectively enhance the performance of our system, achieving scalability, simplicity and reliability.

3

System Development

In this chapter is presented the approach taken in the development of a solution for the automated shopping system to be used in retail using beacon technology. This chapter starts by introducing the system needs and identifying the main problems regarding the implementation of the proposed system. The chapter follows by enumerating several requirements with use cases and proposing an efficient system architecture. The chapter then describes in detail each of the components present in the system architecture to understand the importance and their role in the communication and performance of the system.

3.1 Context Identification

The solution developed aims to fulfil the needs of an automated shopping system in retail integrating beacon technology to provide clients with the ability to walk into a store and proceed with their shopping all through the usage of their own smartphones. Clients then can finish the shopping without going through any sort of checkout since the phone application will automatically bill their purchases and allow them to walk out of the store with a different shopping experience than what we are so familiar with. Retailers are provided with the ability to use personalized marketing with the data collected from users inside the stores, persuade clients with the purchase of products through a dynamic and content engaging user interface where products are displayed, and the information collected from the clients' shopping activities can be used in different new applications like improving client experience by understating the best product placement, etc. Retailers are also provided with a real-time dashboard that can be used by the staff inside their stores to track clients' shopping activity or any other important information about the store's state and events.

3.2 System Requirements

A system of this complexity, to achieve the needs described in the last section 3.1, needs many system features to be implemented and thus defining them with functional and non-functional requirements is essential to understand how the system operates and how these requirements will help the system under development meets the user needs. Before listing the requirements, we need a brief overview of the system to better understand what the following requirements propose. The system uses a three-tier architecture which is described in more detail in the next section of this chapter. This type of architecture has a client tier, a server tier and a data tier. On the client tier, we have the smartphone application that the client will use to receive the signals from the beacons and to provide them with a user interface to perform their shopping in the store. The client tier also contains a web application with dashboard functionalities which is solely used by the staff of the retail store to monitor real-time data and provide the staff with a user interface for displaying different types of information. The server tier is used to do the major computing processes of the system and to share and store resources in the database of the data tier.

3.2.1 Functional Requirements

Functional requirements outline different functionalities and behaviours of the system. There are different components in our system and each component is independent, thus each having a broad list of functional requirements. The functional requirements for the smartphone application are shown in table 3.1, some of the most relevant functional requirements listed are related to the processing of beacon signals, displaying of various shopping activities information and processing of purchases. These requirements hold the highest importance in the implementation of the smartphone application.

Table 3.1: Functional requirements for the smartphone application.

ID	Functional Requirement
1.0	Registration for user creation.
1.1	Authentication for accessing the shopping functionalities.
1.2	Automatic authentication for users with previously validated authentication.
1.3	Scanning of beacon signals.
1.4	Processing of beacon signals.
1.5	Displaying of products.
1.6	Displaying of detailed information about a product.
1.7	Addition of products to a cart.
1.8	Deletion of products from a cart.

1.9	Displaying of products from a cart.
1.10	Purchasing of products.
1.11	Processing of transactions with debit cards.
1.12	Displaying of previously made purchases.
1.13	Displaying of detailed information about a previously made purchase.
1.14	Displaying of a receipt from a previously made purchase.
1.15	Logging out from the application.

The functional requirements for the web application are shown in table 3.2. Some of the most relevant functional requirements listed are related to the displaying of various store information and store management. For the development of the web application, these requirements are of utmost importance.

Table 3.2: Functional requirements for the web application.

ID	Functional Requirement
2.0	Authentication for accessing the dashboard functionalities.
2.1	Automatic authentication of users with previously validated authentication.
2.2	Displaying of store revenue.
2.3	Displaying of store revenue target.
2.4	Listing of users.
2.5	Listing of purchases.
2.6	Listing of products.
2.7	Displaying of detailed information about a product.
2.8	Displaying of detailed information about a user.
2.9	Displaying of detailed information about a purchase.
2.10	Displaying of information in charts.
2.11	Addition of products to the store.
2.12	Displaying of a report about store activity.
2.13	Logging out from the application.

The functional requirements for the server are shown in table 3.3, some of the most relevant functional requirements listed are related to the calculation and processing of indoor localization and processing of information from the whole system. The server is one of the most important components, responsible for performing almost all the computation from the system and implementing these requirements is of utmost importance.

Table 3.3: Functional requirements for the server.

ID	Functional Requirement

3.0	Validating registrations of users.
3.1	Validating authentications of users.
3.2	Encryption of passwords from registrations.
3.3	Preprocessing beacon signal values.
3.4	Calculating localization of users.
3.5	Processing location information of users.
3.6	Processing information of products.
3.7	Processing information of purchases.
3.8	Processing information of users.
3.9	Processing receipts of purchases.
3.10	Processing reports of store activity.

The functional requirements for the database are shown in table 3.4, some of the most relevant functional requirements listed are related to the management of data from the system. These requirements are essential in the implementation of the database for the well functioning of the system.

Table 3.4: Functional requirements for the database.

ID	Functional Requirement
4.0	Modelation of user data.
4.1	Modelation of product data.
4.2	Modelation of purchase data.
4.3	Modelation of location data.
4.4	Management of user data.
4.5	Management of product data.
4.6	Management of purchase data.
4.7	Management of location data.

3.2.2 Non-Functional Requirements

Non-Functional requirements describe the system in an abstract manner, by defining how the system should work in terms of performance, scalability, security, etc. Every component of the system contributes to the performance of the system, thus the listing of the non-functional requirements will take into account the system as a whole instead of listing requirements for each component of the system. Some most relevant requirements are the following:

1. The products displayed for the clients on the smartphone application should be updated in less than 2 seconds after the clients have changed their location.

2. The dashboard should update the information on display in less than 1 second after a shopping activity to satisfy the real-time characteristic of the dashboard.
3. The smartphone application must be supported on different iOS versions from iPhone devices.
4. The localization techniques should not provide an error higher than 1.5 meters in estimating the client location.
5. The system must ensure the security of users' passwords with encryption before storing the credentials of the user in the database.
6. The dashboard must be available to the personnel of the store during business hours.
7. Only authenticated users with the role 'admin' should be able to access the dashboard.
8. Only authenticated users should be able to access the smartphone application.
9. The smartphone application interface has to be user-friendly and easy to use.
10. The dashboard application interface has to be user-friendly and implemented with good visualization tools.

3.2.3 Use Cases

Use cases describe processes that help understand the flow of the system. By conducting use cases it is possible to analyze and organize the requirements as well as the features in a system and identify how they should be implemented. In use cases, actors are users who interact with the system. In the developed system there will be two types of users, client and admin. Table 3.5 lists some relevant use cases for the system.

Table 3.5: List of use cases.

Table	ID	Name	Description
3.6	1.0	Registration	Registration of a client in the system.
3.7	1.1	Products Purchase	Purchase of products by the client in the system.
3.8	1.3	Check Purchases	View past purchases made by the client in the system.
3.9	2.0	Search Users	Search for active users inside the store by the admin.
3.10	2.1	Product Addition	Addition of a new product in the system by the admin.
3.11	3.0	Authentication	Authentication of the user/admin in the system.

3.2.3.1 Registration

The registration use case is necessary to understand how a user creates their credentials to be used in authentication to access the smartphone application. To notify the success of registration in the system, the server needs to validate the registered credentials, the use case flow is described in table 3.6.

Table 3.6: Registration Use Case.

Use Case ID	1.0
Use Case Name	Registration
Actors	Client
Normal Flow	1. Client enters the registration page. 2. System displays a registration form. 3. Client fills out the form with registration details. 4. Client submits the registration. 5. System validates the registration details.
Alternative Flow	5.a System presents an alert of an unsuccessful registration.

3.2.3.2 Products Purchase

The product purchase use case is necessary to understand how a user purchases a product in the system of the store. To make a purchase a client will need to go through various steps which the use case flow describes in table 3.7.

Table 3.7: Products Purchase Use Case.

Use Case ID	1.1
Use Case Name	Products Purchase
Actors	Client
Normal Flow	1. Client enters the shop page. 2. System calculates the user location inside the store. 3. System displays the products available around the user. 4. Client adds the wanted products to the cart. 5. System saves the added products. 6. Client enters the cart page. 7. System displays the cart products. 8. Client proceeds with the payment for the products. 9. System validates the purchase.

Alternative Flow	3.a System has no products available. 4.a Client is not interested in any products available. 5.a System has no product quantity availability. 5.b Client enters the cart page. 5.c System has no cart products. 7.a System has no cart products. 8.a Client modifies the product quantity 8.b Client proceeds with the payment for the products. 8.c System validates the purchase. 9.a System presents an alert of an unsuccessful purchase.

3.2.3.3 Check Purchases

The check purchases use case is necessary to understand how a user can get the details of a previous purchase performed in the system. To see purchase details a client will need to go through various steps which the use case flow describes in table 3.8.

Table 3.8: Check Purchases Use Case.

Use Case ID	1.2
Use Case Name	Check Purchases
Actors	Client
Normal Flow	1. Client enters the account page. 2. System displays account information. 3. Client enter the purchases page. 4. System displays a list of purchases made by the client. 5. Client enter the purchase page. 6. System displays purchase information.
Alternative Flow	4.a System has no purchases made by the client.

3.2.3.4 Search Users Use Case

The search users use case is necessary to understand how the personnel can search on the dashboard for active users from the store. To see active users in the store it will be necessary to go through various steps which the use case flow describes in table 3.9.

Table 3.9: Search Users Use Case.

Use Case ID	2.0

Use Case Name	Search Users
Actors	Admin
Normal Flow	1. Admin enters the dashboard page. 2. System displays the dashboard components. 3. Admin enters the users page. 4. System displays a list with active users in the store. 5. Admin views the users in store.
Alternative Flow	4.a System has no users active in the store.

3.2.3.5 Product Addition Use Case

The product addition use case is necessary to understand how the personnel can add a new product to the store on the dashboard. To add a new product the admin will need to go through various steps which the use case flow describes in table 3.10.

Table 3.10: Product Addition Use Case.

Use Case ID	2.1
Use Case Name	Product Addition
Actors	Admin
Normal Flow	1. Admin enters the dashboard page. 2. System displays the dashboard components. 3. Admin enters the products page. 4. System displays a list with products in the store. 5. Admin enters the product submission page. 6. System displays a product form. 7. Admin fills out the form with product details. 8. System validates the product details.
Alternative Flow	8. System displays an alert of an unsuccessful product submission.

3.2.3.6 Authentication

The authentication use case is necessary to understand how a user or admin can authenticate using their credentials to access the smartphone application or the dashboard. To notify the success of authentication in the system, the server needs to validate the authentication credentials, the use case flow is described in table 3.11.

Table 3.11: Authentication Use Case.

Use Case ID	3.0
Use Case Name	Authentication
Actors	Client & Admin
Normal Flow	1. Client enters the login page. 2. System displays a login form. 3. Client fills out a form with authentication details. 4. Client submits the authentication. 5. System validates the authentication details.
Alternative Flow	5.a System presents an alert of an unsuccessful authentication.

Figure 3.1 showcases the use cases diagram for the user and admin actors of the system in development.

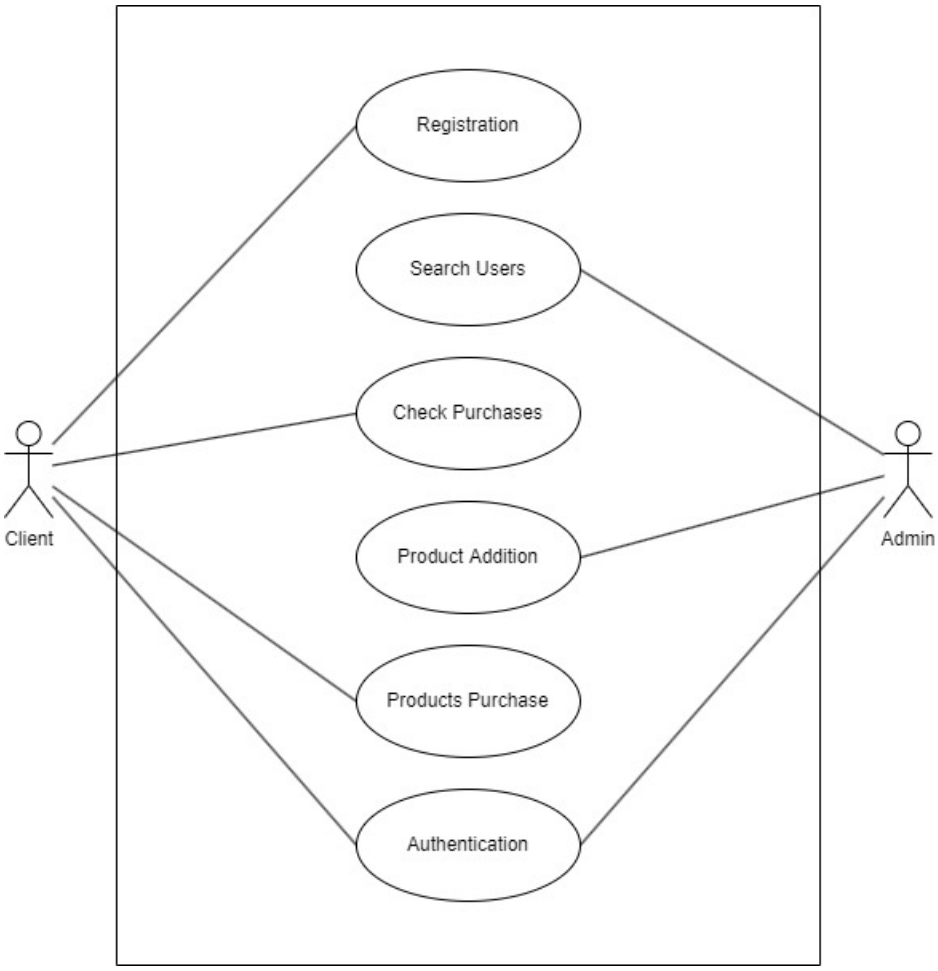


Figure 3.1: Use cases diagram.

3.3 System Architecture

Taking into consideration the objectives and requirements intended for the system discussed in sections 3.1 and 3.2, a three-tier architecture is proposed. The system architecture is presented in figure 3.2 composed of beacons, smartphone application, dashboard, REST API, server and database.

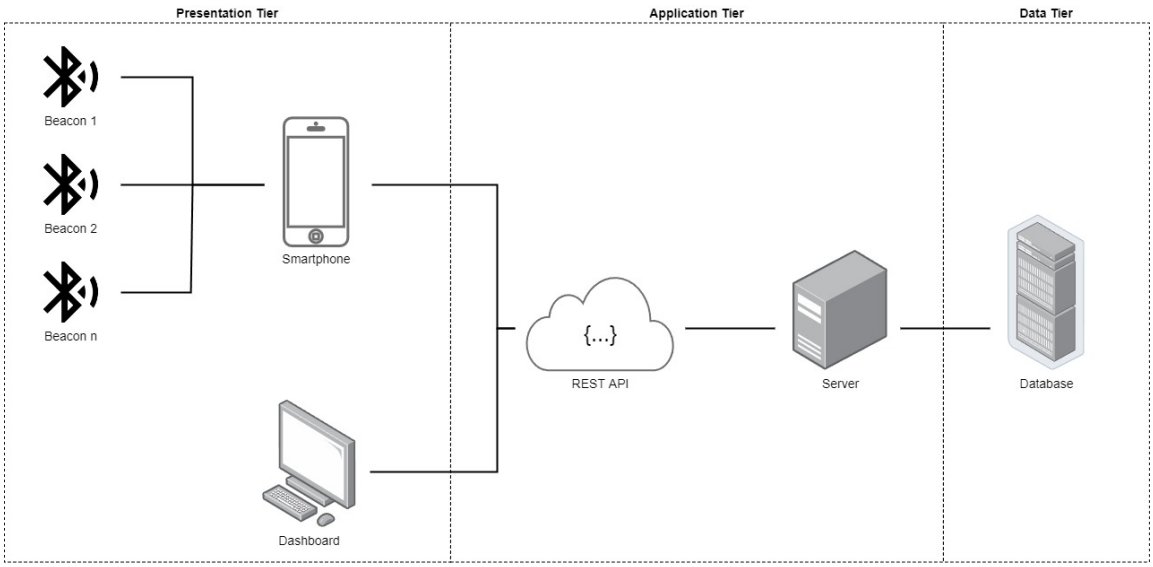


Figure 3.2: System Architecture.

The three-tier architecture is the most common architecture used in distributed systems and follows the client-server model. This architecture splits the complexity of the system into different layers so that we can focus on each component of the system independently. The architecture is organized into three tiers, a **client tier**, an **application tier** and a **data tier**. The client tier is composed of beacons, smartphone applications and a dashboard. The application tier consists of a server and REST API. At last, the data tier is composed of a database.

Beacons are small Bluetooth low-energy devices that will be attached to the ceiling of a retail store. Beacons have only one-way communication meaning that they do not read data. These devices broadcast a constant signal that the smartphone application will receive and process. The beacons implemented in the system use the iBeacon protocol, developed by Apple as discussed in the previous section of the State of Art. Since the smartphone application was chosen to be developed in iOS, integration of this protocol with the developed application was facilitated with the use of existing frameworks provided by Apple for developers. Indoor localization can only be achieved if the smartphone application receives the values of the signals from the beacons. Beacon's main and only function is to communicate with the smartphone application.

Smartphone Application is a software application that will run on clients' smartphones. The smartphone application will serve as a user interface with which the clients will interact while doing their shopping activities inside the retail store. The application is responsible for displaying the products and their information to the client while collecting important data about the shopping activity performed by the client. Another purpose of the smartphone application is to scan for beacon signals and once the application receives these signals, it will use the REST API interface to send the signals values over to the server to be processed. In return, the smartphone application will receive the products based on the client's present location. In brief, the application will communicate with beacons and also with the server through the REST API interface.

Dashboard is a software application that runs on a web browser. The web application will display information about the store's state and events to an admin, a person from the personnel inside the store through charts, tables, etc. The dashboard will be updated with real-time information about the store, mainly active users, recent orders, etc. To display all this information, the application will communicate with the server through the REST API interface to receive the corresponding data.

REST API is an application programming interface used to access the server in a simpler and more flexible way through HTTP requests. The REST API enables the client tier to access the resources of the server. The interface also helps to decouple the client from the server, making each of the components of the system independent from one another.

Server is responsible for providing the resources and data from the store to the client tier. The server also communicates with the database to store and retrieve different types of data. The server does all the computing for the location calculation, the processing of orders, the addition of new products, authentication and is required to be accessible at any time to retrieve the resources needed to make the store functional. The server will communicate with both, the presentation tier and the data tier.

Database is responsible for storing and retrieving different data and information from the store system. The database will define the data models of the large unstructured data sets and allow the server to access and handle the information available.

3.4 Smartphone Application

The **smartphone application** is part of the client tier in the system architecture presented in section 3.3. The application was chosen to be developed for iOS with the SwiftUI framework, a framework written in Swift for building user interfaces for iOS which offers multi-platform development. The framework also presents different components for building appealing user interactive interfaces, components such as lists, buttons, pickers, etc, as well as provides the ability to create custom views with animations which can integrate different types of gestures. Therefore SwiftUI offers a modern way of building user interfaces for iOS with declarative syntax that makes it easy to understand and maintain

while also being fast and efficient. For the development of the smartphone application was used an integrated development environment named XCode, which provides support for Swift and different development tools. Xcode was introduced by Apple for building software for Apple's platforms, including iOS. Xcode has different Apple technologies integrated and provides support for iOS app development making it an ideal choice for those looking to build high-quality iOS apps. The smartphone application can be separated into four independent layers of development as shown in figure 3.3. By doing this, maintainability is increased as each layer has its own logic and can be updated and maintained independently from the other layers. This makes it easier to fix bugs, add new features or make a change in one part of the application without affecting the existing parts. This also promotes a clear and organized codebase, additionally improving scalability and performance. Each layer corresponds to a class inside our application that does not correspond with the business logic of any View inside the user interface but rather makes part of the application as a whole. The beacon detector layer is responsible for discovering beacons and processing the reception of beacon signals. The network layer is responsible for communicating with the server tier of the system architecture. The cart layer is responsible for managing the shopping cart products. And at last, the user interface layer is responsible for presenting users with different user interface interactive components to do their shopping through the smartphone application. Below are listed small descriptions of each technology used in the smartphone application development:

- **Xcode** [27]- An integrated development environment created by Apple, used for development of iOS software for macOS, iPadOS, etc. XCode supports many programming languages, including C, C++, Swift, etc. Xcode comes with a text editor, a compiler and a build system for application development.

- **SwiftUI** [26]- Cross-platform framework, also known as a user interface toolkit which allows the design of front-end applications in a declarative way.

- **Swift** [25]- General-purpose, open source programming language released by Apple. Swift was developed to be used for the development of systems, mobile, desktop apps and cloud services.

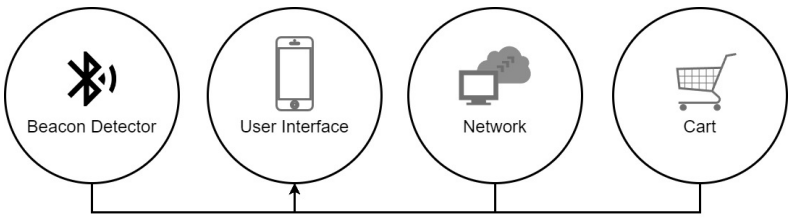


Figure 3.3: Smartphone Application Layers.

3.4.1 Beacon Detector

All the communication between the beacons and the smartphone application is processed by the beacon detector layer which corresponds to the BeaconDetector class inside our application. By processing all the logic related to monitoring and the retrieval of beacons inside a separate class, we can decouple the main functionalities of our smartphone application and work on the beacon detection individually. To detect the beacons the smartphone application uses the iBeacon protocol.

3.4.1.1 Location Privacy

Location is a service available on smartphones which are considered to be private to each user and Apple forces this privacy as well, being up to the user to allow the detection of their location. Thus to allow the iOS to monitor for beacons, the application needs the user's permission to allow the monitoring.

3.4.1.2 Implementation

To implement the detection of beacons, the smartphone application has to first request location access authorization from the user, a mandatory step for the application to be allowed to follow with the scanning for beacons and receiving their corresponding signal values. The beacons implemented in the system use the iBeacon protocol from Apple. Apple provides iOS applications with a framework to communicate with iBeacons, the Core Location framework. The Core Location[8] framework gathers data from Bluetooth to provide services like retrieving the position of the smartphone device relative to an iBeacon device or providing information about the iBeacons detected. The framework lets us configure how to notify the user about the location and also deliver the location updates.

3.4.1.3 Beacon Detection

To detect beacons, the smartphone application firstly creates a CLLocationManager object from the Core Location Framework and sets itself as a delegate to conform to the protocol. By doing this, an authorization request can then be sent with the requestAlwaysAuthorization() method from the object. After the authorization has been granted by the user, the delegate object gets notified by the locationManagerDidChangeAuthorization() method that the authorization status has changed and if the authorization has been granted then the scanning for beacons can start. To start the detection of beacons, a CLBeaconRegion object is created to identify a beacon uniquely or to identify a group of beacons with the same UUID that is wanted to monitor. Section 2.3.1 covers the identification standard of an iBeacon. This region object is passed to the CLLocationManager by calling the startMonitoring and startRangingBeacons methods, allowing the iOS application to monitor for beacons. The ranging method from CLLocationManager updates the

location by retrieving an array of beacons it finds for the given defined region. Figure 3.4 shows the iBeacons detection activity diagram.

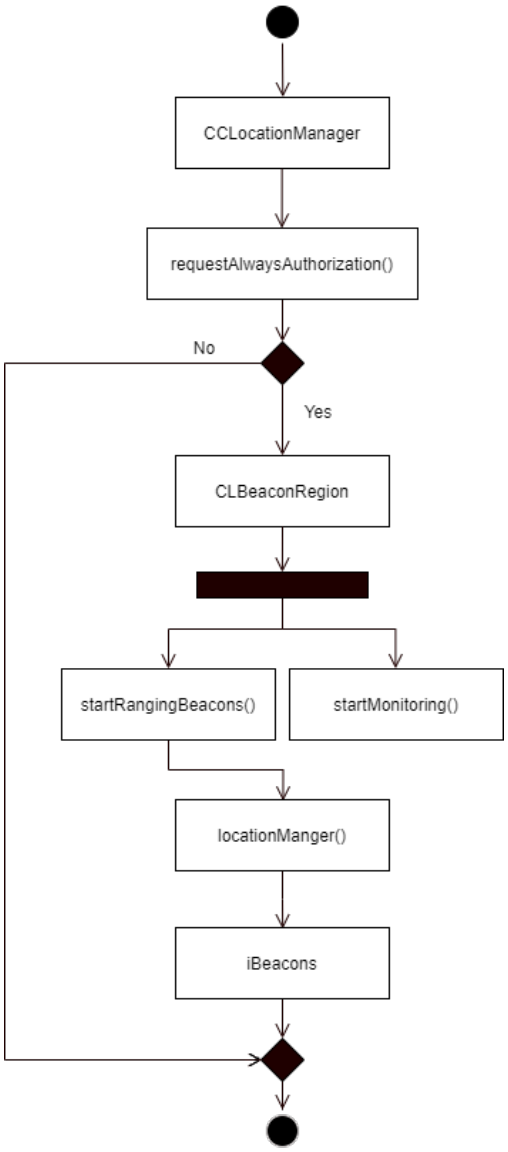


Figure 3.4: Beacon detection activity diagram.

3.4.2 Network

The Network layer of the smartphone application is entrusted with handling all the communication with the server tier which corresponds to the Network class inside the smartphone application. By having the processing of requests to the server tier inside a separate class it allows us to work on the smartphone communication individually. The communication between the smartphone application and the server is made through HTTP requests.

3.4.2.1 HTTP Requests

A client sends an HTTP request to a host located on the server in order to access a resource on a server. The request contains an URL which provides information about the resource the client is trying to access on the server. The server proceeds with answering the request with an HTTP response which contains the resource requested by the client and a status code that informs the client if the request was successful or denied.

3.4.2.2 Cache

In smartphones, the cache is a memory storage which stores temporarily files of data that are likely to be needed in the near future. This way smartphones can quickly access the temporary data and retrieve the associated information needed boosting the data retrieval process by not accessing slower storage layers.

3.4.2.3 Implementation

To implement the communication of the smartphone with the server, the application provides different requests to access different resources on the server. Every request for data to the server tier is independent thus the Network class defines a method for each request since each of the methods has differently defined URLs, HTTP methods, headers and body data for the HTTP request. The system only uses JSON format data in the communication within the system. The application uses the Foundation framework. The Foundation[18] framework provides a base layer of functionality which includes networking. Also, the user interface of the smartphone application displays a great number of images from the products of the store. Requesting and processing so many images from the server responses can cost performance. Thus to help with this, the memory cache is used to store recent requested images from the server. Reusing the images stored in the cache provides some performance benefits since the requesting of the images doesn't need to be performed again.

3.4.2.4 Requests

To implement the requests for the Network class, the smartphone application firstly creates a URLRequest object where the URL, HTTP method, etc, fields are defined. If there is any data to be sent inside the body of the HTTP request, a JSONSerialization object is used to serialise an object and convert it to a JSON object. After the preparation of the request, the URLSession object is created and the URLRequest object is passed inside. This object allows access to the network data transfer tasks. It is possible to receive the results with a completion handler available with the URLSession object. For that, a data task - DataTask object - is created which delivers the server's response, data and any possible errors which we can then handle. If the response is expected to contain some type of data, the DispatchQueue object is used to process the body from the response

asynchronously. At last, the resume method is called to proceed with the request to the server. Figure 3.5 shows the HTTP activity diagram and the listing 1 showcases a partial part of the code related to request creation.

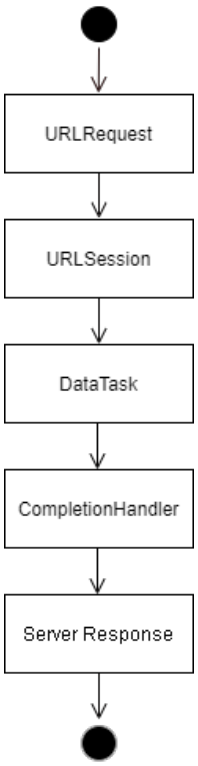


Figure 3.5: HTTP activity diagram.

3.4.2.5 Caching

To allow caching of data files like images, the smartphone application creates a NSCache object. The NSCache class is a mutable collection which stores temporarily key-value data that gets to be deleted when the resources of the device are low. To store a data file like an image in the NSCache object, the setObject() method is used where the object to be stored in the cache and a key to associate with the object are passed as parameters. To retrieve a data file, the object() method is called from the NSCache object which retrieves the object given the key associated with the object passed as a parameter. Listing 2 showcases a partial part of the code related to caching of the images.

3.4.3 Cart

The cart layer is responsible for the management of the shopping cart of the user which corresponds to the CartItems Class inside the smartphone application. By having the logic of managing the products of the shopping cart inside a separate class we can decouple

```
class Network: NSObject, ObservableObject, UIDocumentInteractionControllerDelegate {
    //...
    func sendBeaconValues(beacons: [Beacon])
    {
          //...
          let jsonEnconder = JSONEncoder()
          let jsonData = try? jsonEnconder.encode(beacons)
          var request = URLRequest(url: url)
          request.httpMethod = "POST"
          request.httpBody = jsonData
          request.setValue("application/json", forHTTPHeaderField: "Content-Type")
          request.setValue(UserDefaults.standard.string(forKey: "token"),
          forHTTPHeaderField: "x-access-token")
          let task = URLSession.shared.dataTask(with: request) {
          (data, response, error) in
              //...
              if response.statusCode == 200 {
                  guard let data = data else
                  {
                      print("sendBeaconValues Error: 3")
                      return
                  }
              }
              //...
              DispatchQueue.main.async {
                  do{
                      self.section = try JSONDecoder().decode(Section.self, from: data!)
                      if(self.section.section == "0")
                      {
                          return
                      }
                      self.getShopItems(section: self.section.section)
                  }
                  //...
              }
          }
          task.resume()
    }
    //...
}
```

Listing 1: Smartphone Application - Request

the main functionalities of our smartphone application and work on the cart system individually.

```
class Network: NSObject, ObservableObject, UIDocumentInteractionControllerDelegate {
    let imageCache = NSCache<NSString, UIImage>()
    //...
    func loadImageUsingCache(withUrl urlString : String) {
        //...
        if let cachedImage = imageCache.object(forKey: urlString as NSString)  {
            self.itemImage = cachedImage
            self.loading = false
            return
        }

        URLSession.shared.dataTask(with: url!, completionHandler: {
        (data, response, error) in
            //...
            DispatchQueue.main.async {
                if let image = UIImage(data: data!) {
                    imageCache.setObject(image, forKey: urlString as NSString)
                    self.itemImage = image
                    self.loading = false
                }
            }
        }).resume()
    }
    //...
}
```

Listing 2: Smartphone Application - Caching

3.4.3.1 Cart System

A part of e-commerce software on a web or smartphone application that allows clients to select and store products for eventual purchase.

3.4.3.2 Implementation

To implement the cart system, the smartphone application provides different methods for cart management. All these methods from the CartItems class are called from the user interface view components.

3.4.3.3 Cart Mangament

The class contains the addItemToCart() method, which adds and stores an item inside the shopping cart, a deleteItemFromCart() method, which deletes and empties an item from the shopping cart, a getItemsFromCart() method, which returns an array of all the items added to the shopping cart and a resetCart() method, which empties the shopping cart of the user.

3.4.4 User Interface

This layer is responsible for the development of the user interface for the application. By having the development of the user interface inside a group of separate classes we can decouple the main functionalities of our smartphone application and work on the user interface individually. The User Interface Layer interacts directly with the beacon detector layer, the network layer and the cart layer. The interface is the point at which users, clients of the retail store, interact with the application to do their shopping inside the store thus the goal of this layer is to build an effective UI for the application to make the user's experience appealing, easy and intuitive.

3.4.4.1 MVVM Pattern

In the Model-View-ViewModel (MVVM) pattern, the user interface is divided into three main parts, Views, Models, ViewModels. The Views are responsible for creating all the UI components and presenting them to the user on the screen. The Models hold all the data that is to be shown on the screen inside the UI components. And the ViewModels are responsible for all the business logic and processing of the data from the models in order to display them on screen. Figure 3.6 showcases the MVVM pattern diagram.

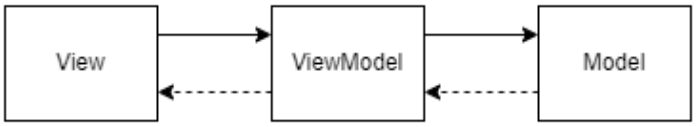


Figure 3.6: Application MVVM Pattern.

3.4.4.2 Implementation

When Apple was creating the framework, the MVVM pattern was selected as the main pattern for their framework and adopted some clean solutions to support the pattern. Thus the MVVM was the design pattern chosen for the development of the smartphone application because of its intuitive use within the SwiftUI framework as well as the separation that the design pattern provides between the view from the business logic. The views developed were based on similar smartphone applications and followed the requirements specified in section 3.2. By using popular design patterns for the UI elements, the application provides simplicity since users can rely on previous experience while interacting with this application. Table 3.12 lists the relevant views from the application.

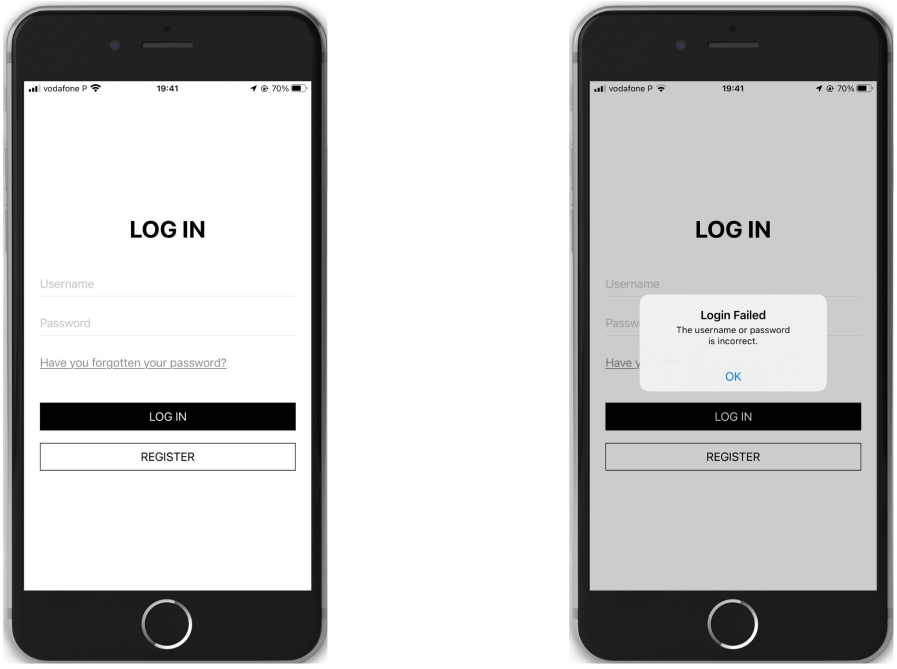
Table 3.12: User Interface Views.

Section	Name	Description
3.4.4.3	LoginView	Authentication for access to the application.

3.4.4.4	RegisterView	Registration for creating a user and authentication details.
3.4.4.5	MainPageView	Display of the bottom tab bar navigation.
3.4.4.7	ProductsListView	Displays the products available around the user.
3.4.4.8	ProductView	Displays the information of a certain product.
3.4.4.6	CartView	Displays the products added in the cart.
3.4.4.9	PaymentView	Insertion of card details for payement.
3.4.4.10	ProfileView	Access to account information.
3.4.4.11	MyPurchasesView	Displays the purchases done by the user.
3.4.4.12	MyPurchaseView	Displays the information of a purchase done by the user.

3.4.4.3 Login

The view corresponding to the login page is the LoginView. This is the first view which appears when the application is opened on the smartphone. This view has a small form with two inputs, one input for the username and one input for the password. If the inputs are tapped, a text pad appears for the user to text the corresponding details. There are also two buttons, one button for transitioning into the registration view and one for requesting authentication with the details entered in the form. If the user authentication is successful, the user gains access to the application and is transitioned into the main page view. In case the user authentication fails, an alert card will pop up alerting the user of the succeeding error. Figure 3.7 shows the interface of the login view.



(a) (b)

Figure 3.7: a) Login view b) Login view with an error alert.

3.4.4.4 Register

The view corresponding to the registration page is the RegisterView. The view contains a back button to return to the login view. This view displays a form with two inputs, one input for the username and one input for the password. If the inputs are tapped, a text pad appears for the user to text the corresponding details. There is also a checkbox input for signing the confirmation of reading the terms and agreement and a button for requesting the registration of the user. In case the registration fails or the checkbox isn't signed, an alert card will pop up alerting the user of the corresponding succeeding error. In case of successful registration, the application automatically returns to the login view. Figure 3.8 a) and b) show the interface of the register view.

3.4.4.5 MainPageView

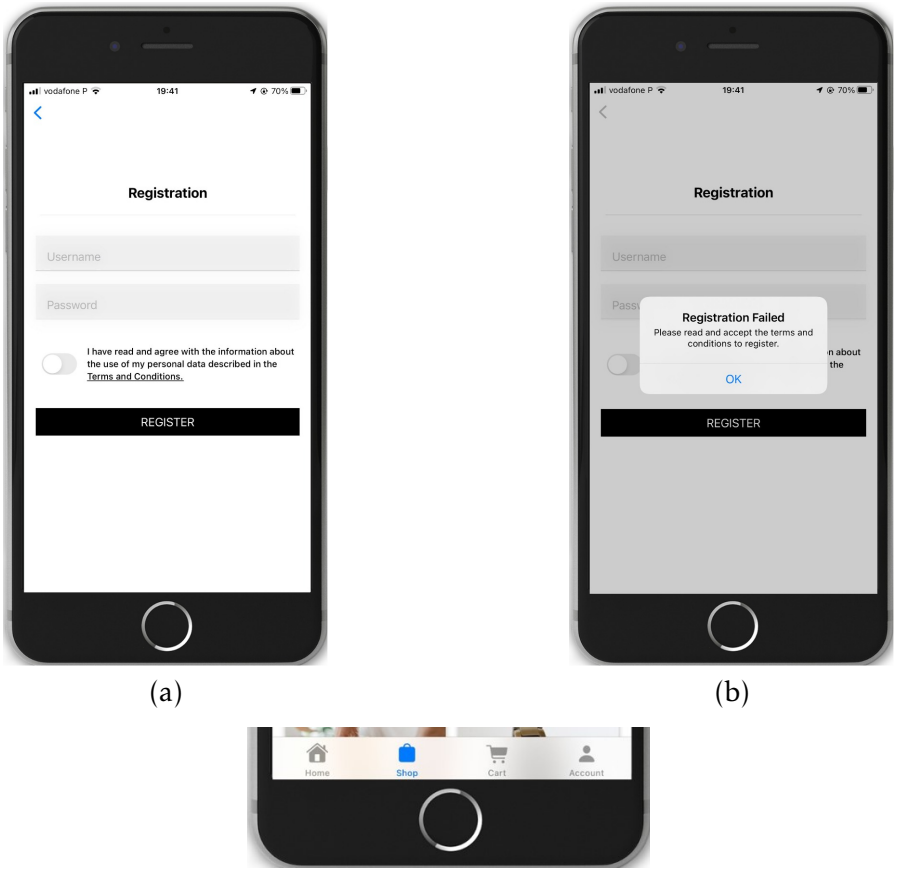
The view corresponding to the main page is the MainPageView. The view contains the bottom tab bar navigation which contains four main tab items, home, shop, cart and account. By tapping each of the tab items on the bottom tab bar navigation, the view displays a view corresponding to that tab item. The home tab item corresponds to the HomepageView, the shop tab item corresponds to the ProductsListView, the cart tab item corresponds to the CartView and the account tab item corresponds to the ProfileView. Figure 3.8 c) shows the bottom bar tab navigation view.

3.4.4.6 CartView

The view corresponding to the cart page is the CartView. This view contains a scrollable view which displays a list of products that are inside the cart system. Each product is displayed with an image, two circular plus and minus buttons that if tapped, increase or decrease the quantity of the corresponding product inside the cart system, and some product information: name, price, quantity in the cart. By scrolling down the scrollable view with a sliding-down finger gesture, the view showcases all the products in the cart system. By tapping on the image of any of the products displayed, the view is transitioned into the ShopItemView automatically. Below the scrollable view is displayed the total value of the cart products and a checkout button. By tapping the checkout button, the view is transitioned into the PaymentView automatically. If the cart system has no products added, the scrollable view will be empty and no total value or button is displayed. Figure 3.9 b) shows the cart view.

3.4.4.7 ProductsListView

The view corresponding to the shop or the products list page is the ProductsListView. The view contains a scrollable view which displays a dynamic grid consisting of cards with some product information: name, price, image. By scrolling down the scrollable view with a sliding-down finger gesture, the view showcases all the products available



(a) (b)

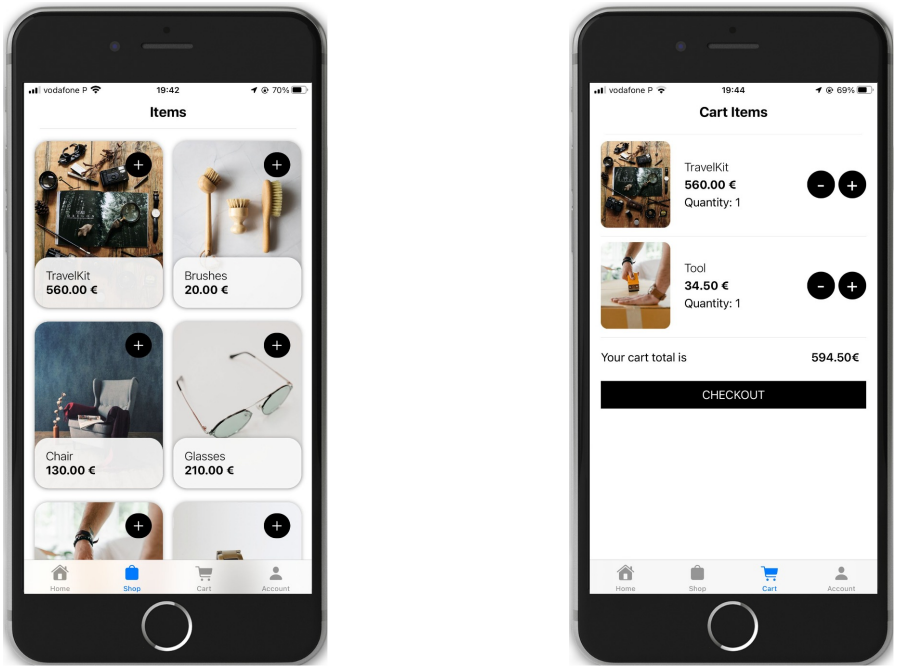
(c)

Figure 3.8: a) Register view. b) Register view with an error alert. c) Bottom tab bar navigation.

in the store. Each product card has also a circular plus button which adds the product to the cart system of the smartphone application. By tapping a product card, the view is transitioned into the ShopItemView automatically. Figure 3.9 a) shows the product list view.

3.4.4.8 ProductView

The view corresponding to the product display page is the ProductView. This view contains an image of the product and a sliding card from the bottom of the view. The card can be pushed up and expanded by tapping on the top part of the sliding card and making a sliding-up finger gesture. Inside the card, there is a scrollable view which displays product information: name, price, description, quantity available, reference number. Below the scrollable view inside the card, there is also a button which if clicked, adds the product on display to the cart system of the smartphone application. The view contains a back button to return to the products list view or the cart view, depending on where the view was transitioned to. Figure 3.10 shows the product view.



(a) (b)

Figure 3.9: a) Product list view. b) Cart view.

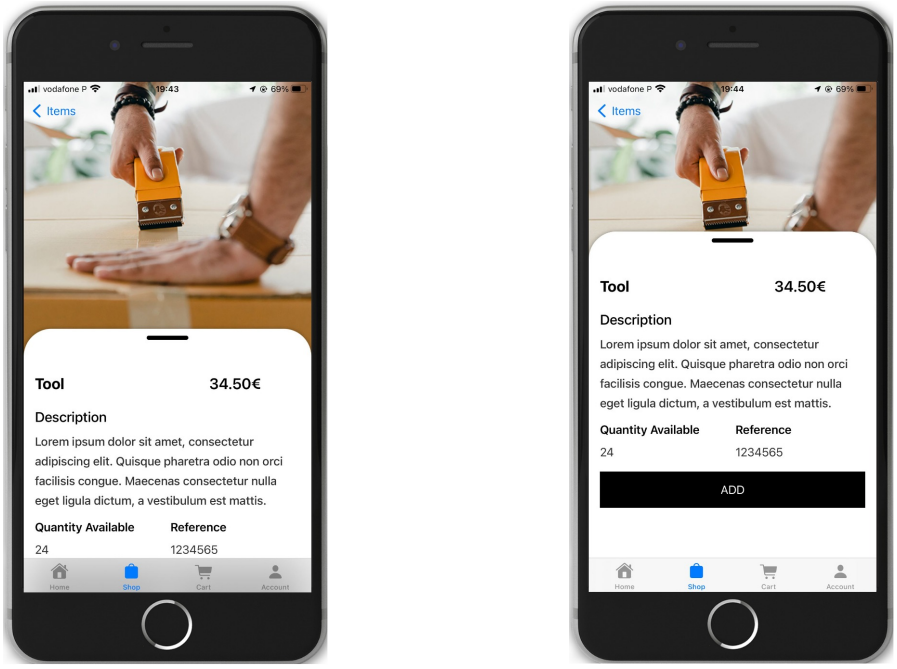


Figure 3.10: Product view.

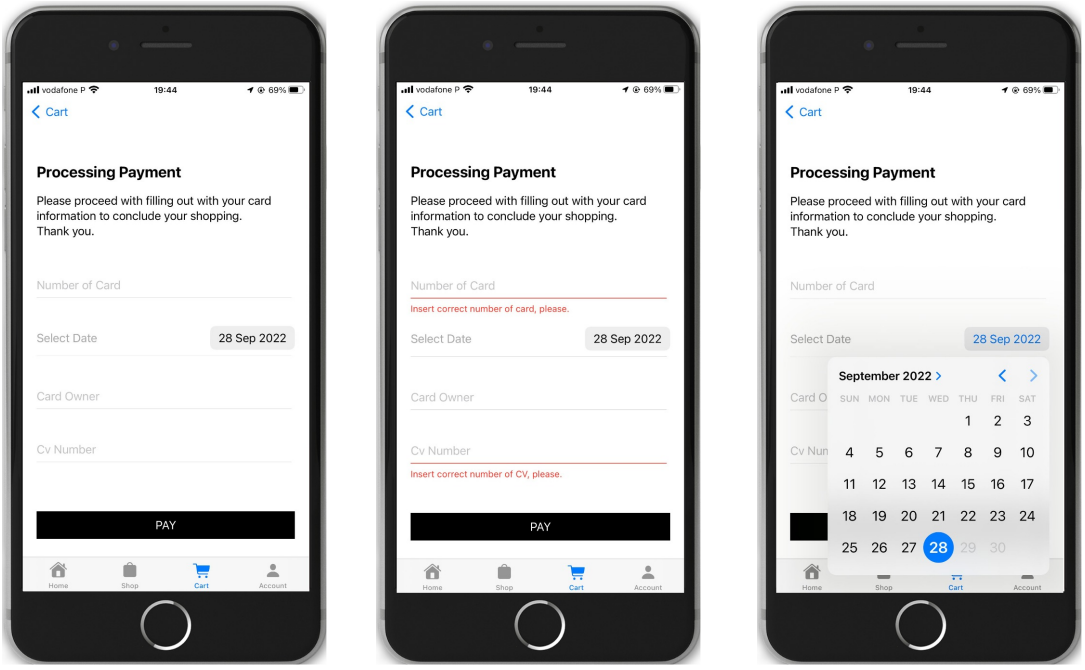
3.4.4.9 PaymentView

The view corresponding to the payment or checkout page is the PaymentView. This view contains a form with four inputs, one input for the card holder's name, one input for

the card's validity date, one input for the card's number and one input for the card's Card Verification Value (CVV) number. If the cardholder's name input is tapped, a text pad appears for the user to text the corresponding details. If the card's number and the card's CVV number inputs are tapped, a number pad appears for the user to text the corresponding details. If the card's validity date input is tapped, a date picker appears for the user to select the corresponding date. There is also a Pay button for requesting the order with the card details entered in the form and the products in the cart system. If the card details are not valid, the information regarding of the form input where the validation failed, will appear in red. If the request proceeds, a confirmation view appears and the application automatically returns to the cart view. Figure 3.11 shows the payment view.

3.4.4.10 ProfileView

The view corresponding to the account page is the ProfileView. This view contains a list of options for accessing account information: purchases, privacy policy, customer service and logout. Each option from the list also acts as a button that can be tapped and transitions to the corresponding view of the option chosen. Tapping the purchases button transitions into the MyPurchasesView. Tapping the logout button transitions into the LoginView. Figure 3.12 a) shows the profile view.

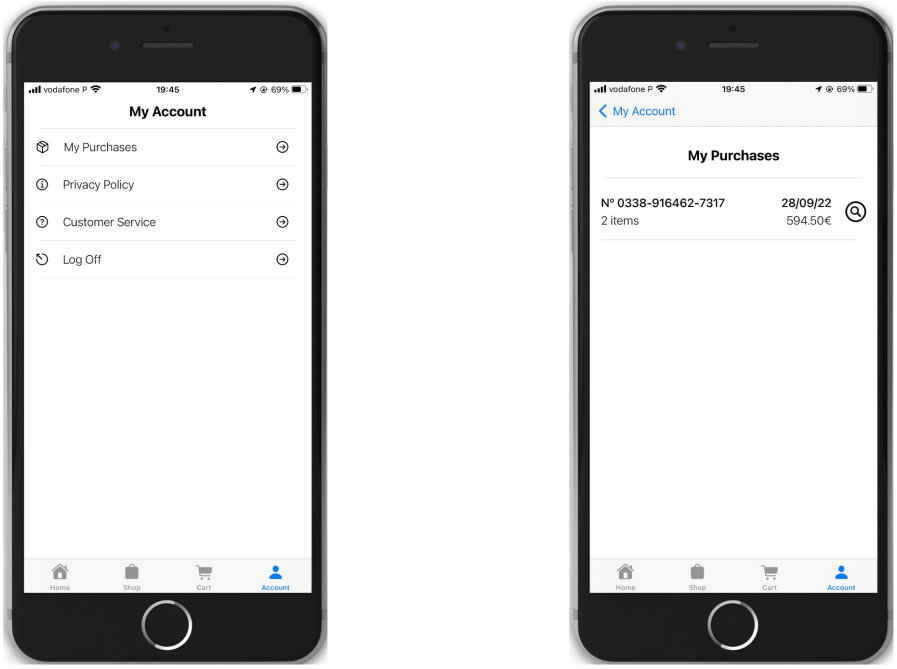


(a) (b) (c)

Figure 3.11: a) Payment view. b) Payment view with errors alert. c) Payment with the date picker.

3.4.4.11 MyPurchasesView

The view corresponding to the list of purchases is the MyPurchasesView. This view contains a list of purchases done by the user. Each purchase displayed from the list also acts as a button that can be tapped and transitioned to the corresponding view. Each purchase from the list is displayed in the view with some part of the purchase's information: order number, date of order, total value, number of products ordered. Figure 3.12 b) shows the purchases view.



(a) (b)

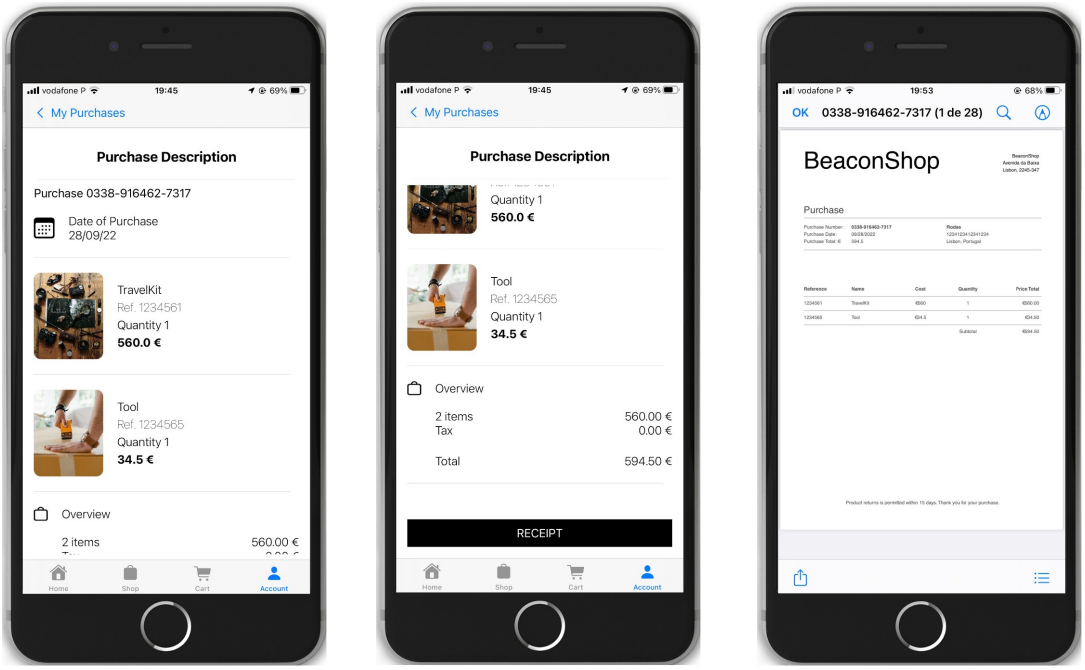
Figure 3.12: a) Profile view. b) Purchases view.

3.4.4.12 MyPurchaseView

The view corresponding to the purchase page is the MyPurchaseView. This view contains a scrollable view which displays the purchase information: order number, date of order, number of products, total value and products from the purchase. By scrolling down the scrollable view with a sliding-down finger gesture, the view showcases all the information about the purchase made. Below the Scrollable view there is a receipt button which if clicked, the smartphone application sends a request to the server to receive a receipt in PDF file format of the purchase. Figure 3.13 a) shows the purchase view.

3.5 Web Application

The **dashboard** is part of the client tier in the system architecture presented in section 3.3. The Dashboard is a web application which was chosen to be developed with the



(a) (b) (c)

Figure 3.13: a) & b) Purchase view c) Receipt in PDF file format.

React Library, an open-source library written in JavaScript. React offers flexibility in the development of component-based client-side web applications where the components can change state and data without reloading the page which provides a significant performance boost. For the development of the dashboard was used a streamlined code editor, Visual Studio Code, which provides excellent support for JavaScript and has a lighting fast source code editor, ideal for development. The web application can be separated into two independent layers of development. The Network layer is responsible for communication with the server tier of the system architecture. The user interface layer is responsible for presenting an admin with different interface interactive components to visualize the state and events of the store through the web application. Below is listed a small description of each technology used in the dashboard development:

- **Visual Studio Code**[75] - A powerful source code editor which runs on desktop, available for Windows, macOS and Linux. Comes with built-in support for JavaScript, TypeScript and Node.js. The editor includes a rich ecosystem of extensions that provide languages, debuggers, and tools to support the development workflow.

- **JavaScript**[28] - Lightweight, interpreted compiled programming language with first-class functions. JavaScript is a very popular scripting language used by browser environments and non-browser environments like Node.js and Adobe Acrobat. The

programming language is considered to be single-threaded, prototype-based, dynamic and multi-paradigm which supports object-oriented, declarative and imperative programming styles.

- **HTML**[22] - HyperText Markup Language which describes the structure of Web pages. Allows the creation and structure of sections, paragraphs, links, etc, using HTML elements, building blocks of a web page) such as tags and attributes.
- **CSS**[9] - Cascading Style Sheets is a stylesheet language used to provide a description of a document written in HTML. CSS dictates how the elements of the Web page should be rendered on a browser.
- **SASS**[66] - Syntactically Awesome Style Sheets is an extension of CSS which provides additional features for CSS like variables, nested rules, inline imports, etc. SASS helps keep CSS organized and allows the creation of style sheets faster.
- **React**[61] - Open-source front-end JavaScript library for building user interfaces with UI components. React uses Virtual DOM which compares the components' previous state and updates only the items in the Real DOM that were changed.

3.5.1 Application Libraries

A library is a collection of reusable prewritten code snippets that provide solutions to specific features. The collection can include network functionalities, user interfaces, etc. In the web application, several different libraries are used to make the development of the application more efficient by means of reducing the time and effort of developing already existing components that can be imported from libraries. Most of the libraries used in the application provide components for the user interface of the application. Below are listed the most relevant libraries used in this application:

- **@mui/icons-material**[40] - SvgIcon components converted from Google Material Icons.
- **@mui/material**[53] - Components library that feature Google's Material Design System.
- **@mui/x-data-grid**[62] - Fast and extendable react data grid component.
- **moment**[42] - Helps with simple parsing, validating, manipulating and displaying of the date/time in JavaScript.
- **react-circular-progressbar**[58] - Circular progressbar component, built with SVG and extensively customizable.
- **recharts**[64] - Composable charting library built on React components.

- **react-router-dom**[63] - A lightweight, fully-featured routing library for the React JavaScript library.

3.5.2 Network

The Network Layer of the web application is entrusted with handling all the communication with the server tier which corresponds to the networkcalls file inside the web application. By having the processing of requests to the server tier inside a separate file it allows us to work on the dashboard communication individually. The communication between the web application and the server is made through HTTP requests.

3.5.2.1 HTTP Requests

A client sends an HTTP request to a host located on the server in order to access a resource on a server. The request contains an URL which provides information about the resource the client is trying to access on the server. The server proceeds with answering the request with an HTTP response which contains the resource requested by the client and a status code that informs the client if the request was successful or denied.

3.5.2.2 Real-time Processing

A real-time application uses a method known as real-time processing which involves processing data practically instantly. This approach doesn't involve any waiting or pausing. As soon as data is input into the system, it is processed and given to the application to be output. Real-time processing typically needs a constant flow of data because of this.

3.5.2.3 Implementation

To implement the communication of the dashboard with the server, the web application provides different requests to access different resources on the server. Every request for data to the server tier is independent thus the networkcalls file has defined a method for each request since each of the methods has differently defined, URLs, HTTP methods, headers and body data for the HTTP request. The application uses the Fetch API, a promise-based interface for fetching resources which uses HTTP requests. Real-time processing is achieved by implementing an object that allows the execution of the HTTP request at specified time intervals.

3.5.2.4 Requests

To implement the requests for the networkcalls file, the web application defines an async method for each different request. Each method receives a state setter object as a parameter for changing the value of the state variable from the useState Hook, functionality from the React Library. A component from the web application uses the useState hook functionality to update the state of the data inside the component. To send a request,

the component calls the corresponding method from the networkcalls file. The method called, uses the fetch() method from the Fetch API which accepts two parameters, resource and options. The resource parameter is assigned an URL string and the options parameter is assigned an object containing custom settings to be applied to the request: method, headers, body, etc. When the answer from the server is received with the queried resource, the request method from the networkcalls file uses the parameter variable to update the data state of the corresponding component. If any type of data is needed to be sent with the request, the method from networkcalls file also receives an additional parameter which is a data object that is parsed to JSON format using the JSON object and its parse() method.

3.5.2.5 Realtime

To implement the real-time processing in the dashboard, the web application uses the window object from javascript with the setInterval() method. This method allows the continuous repetition of the execution of a specified function. The function is passed in the parameters of the setInterval() method and executes HTTP request methods from the networkcalls files to provide the dashboard with the data related to products, purchases and users. Since this is done repeatedly the information is always updated on the dashboard.

3.5.3 User Interface

This layer is responsible for the development of the user interface for the application. By focusing on the development of the user interface on a separate group of files we can decouple the main functionalities of our web application and work on the user interface individually. The user Interface Layer interacts with the Network Layer. The interface is the point at which, the admin, a worker from the staff of the retail store, interacts with the application to visualize the store's state and events thus the goal of this layer is to build an effective UI for the application which provides clear and relevant business data points.

3.5.3.1 Implementation

The web application was built using the React library. The pages from the application were developed based on similar web applications and followed the requirements specified in section 3.2. By using popular UI elements, the application provides simplicity since users can rely on previous experience while interacting with this application. Table 3.13 lists the relevant pages in the application.

Table 3.13: Dashboard Interface Pages.

Section	Name	Description

3.5.3.2	Home Page	Displays important aggregated information.
3.5.3.3	Users Page	Displays the users of the store.
3.5.3.4	Products Page	Displays the products available in the store.
3.5.3.5	Purchases Page	Displays the purchases performed in the store.
3.5.3.6	Single Page	Displays detailed information from users, products or purchases.
3.5.3.7	New Product Page	Provides a form for creating a new product in the store.

3.5.3.2 Home Page

The Home Page corresponds to the main page of the dashboard. This is the front page of the dashboard application also known as the index file as it is the web server directory index which first appears when the application is accessed. This page contains a side menu bar with several buttons displayed in column style to access the rest of the pages of the dashboard. The top bar menu also displays several icons and a search bar. The most relevant icon is the report button/icon which if clicked, downloads a PDF document that contains a report with information about the store activity. The main page displays several important aggregated information through widgets, charts and lists which include users, products, purchases and revenue information with comparative indicators like percentages, progress bars, and pointers. A relevant component of the main page is the feature widget includes information about the target value of daily revenue, weekly revenue and monthly revenue. The feature widget also includes a button on the top right corner that is used to open a small modal box to set a new target value for the daily revenue. In case the new target value is set, all the comparative indicators automatically update. Figure 3.14 shows the home page interface.

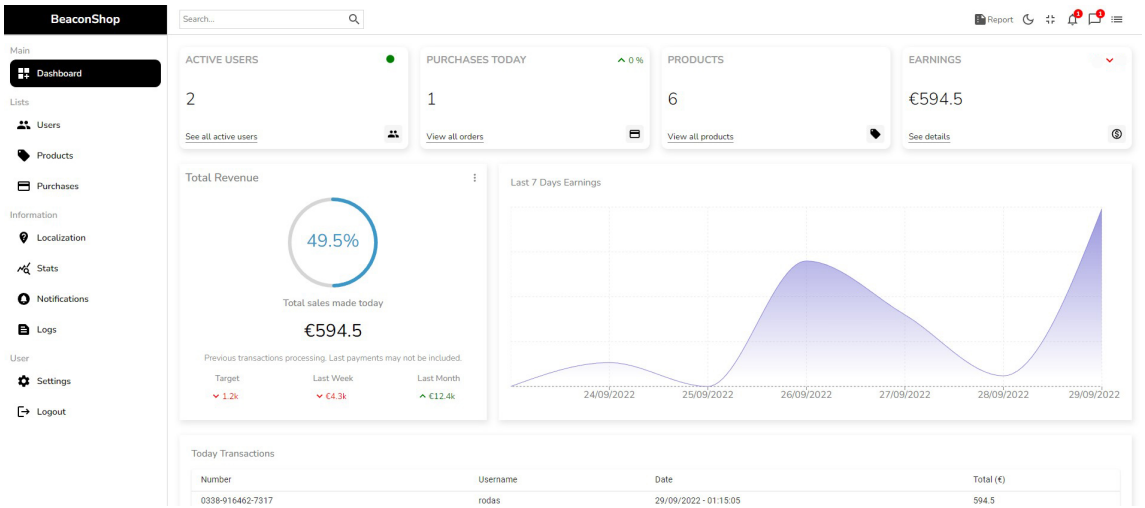


Figure 3.14: Dashboard page.

3.5.3.3 Users Page

The users page refers to the listing of clients of the store. This page contains a datagrid table with pagination where each row corresponds to a client of the store. The table displays the username, status and action columns. The action column has a button for each row which reroutes the dashboard to a single page where detailed information about the user of the row clicked is displayed. Figure 3.15 shows the users page interface.

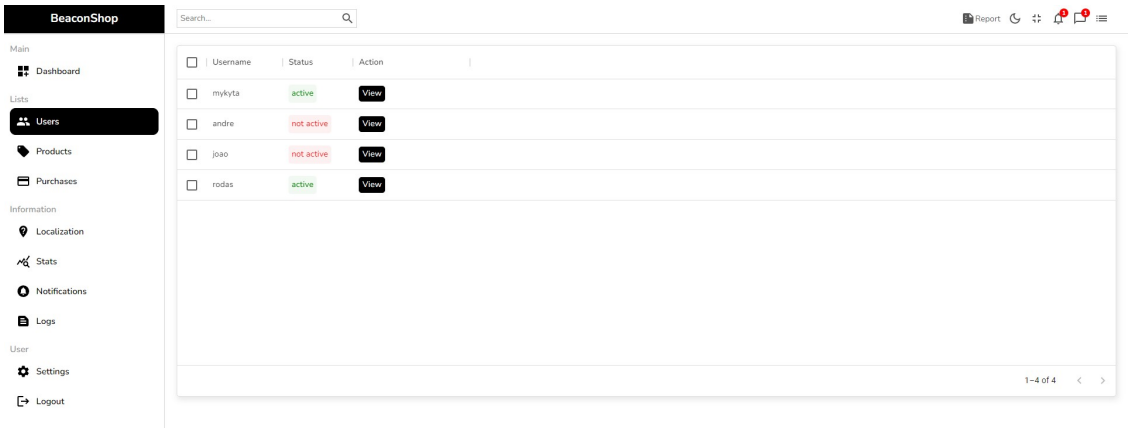


Figure 3.15: Users page.

3.5.3.4 Products Page

The products page refers to the listing of products in the store. This page contains a datagrid table with pagination where each row corresponds to a product in the store. The table displays the reference, name, description, quantity, price, section and action columns. The action column has a button for each row which reroutes the dashboard to a single page where detailed information about a product of the row clicked is displayed. The datagrid of the products page also contains a button on the top right part of the datagrid which reroutes the dashboard to the new product page. Figure 3.15 shows the products page interface.

3.5.3.5 Purchases Page

The purchases page refers to the listing of purchases made in the store. This page contains a datagrid table with pagination where each row corresponds to a purchase made in the store. The table displays the number, username, date, total and action columns. The action column has a button for each row which reroutes the dashboard to a single page where detailed information about a purchase of the row clicked is displayed. Figure 3.17 shows the purchases page interface.

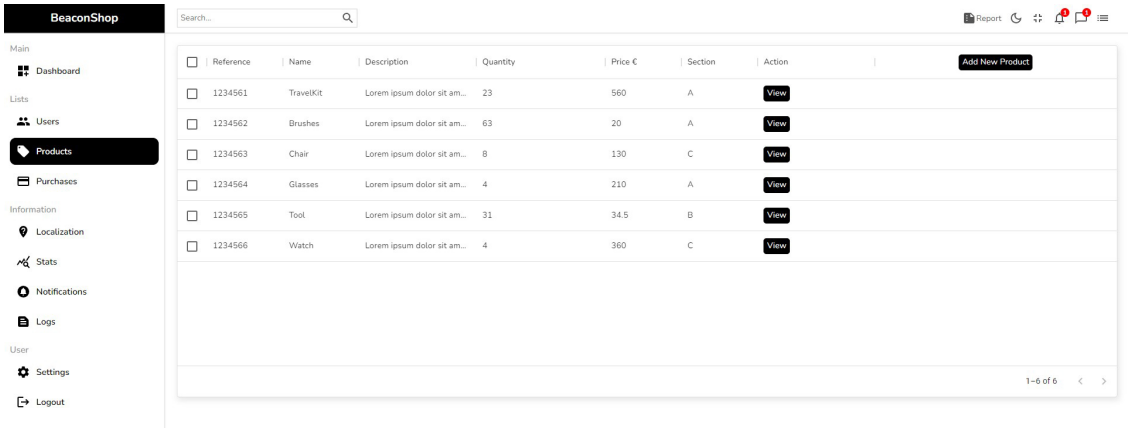


Figure 3.16: Products page.

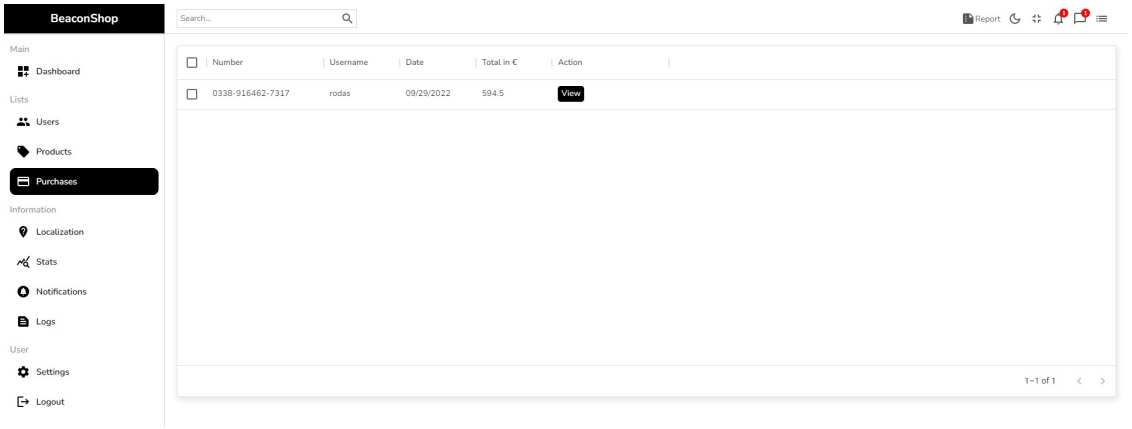


Figure 3.17: Purchases page.

3.5.3.6 Single Page

The Single Page refers to the display of detailed information about users, products or purchases. This page contains a small description area, a chart area and a listing table. Each of these components displays different data information depending on what resource the dashboard rerouted to. For instance, if the dashboard is rerouted to a single page related to a user, the chart will display the previous week's purchases of the user whereas if the dashboard is rerouted to a single page related to a product, the chart will display the previous week's purchases of the product. Figure 3.18 shows the single page interface of a user.

3.5.3.7 New Product Page

The New Product Page refers to the addition of a new product in the store. This page contains a form with several input boxes for the admin to fill out and an image input button to upload the image of the product to be created in the store. Figure 3.19 shows the new product page interface.

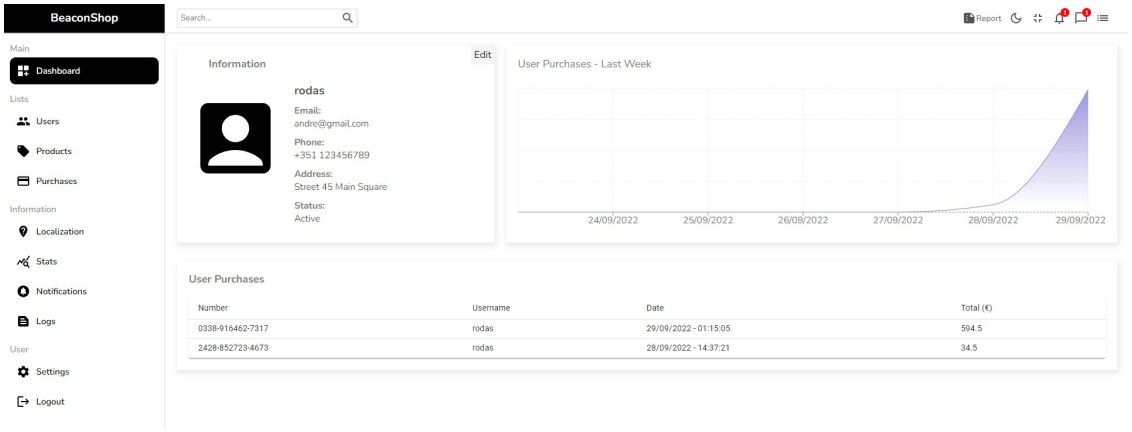


Figure 3.18: Single page.

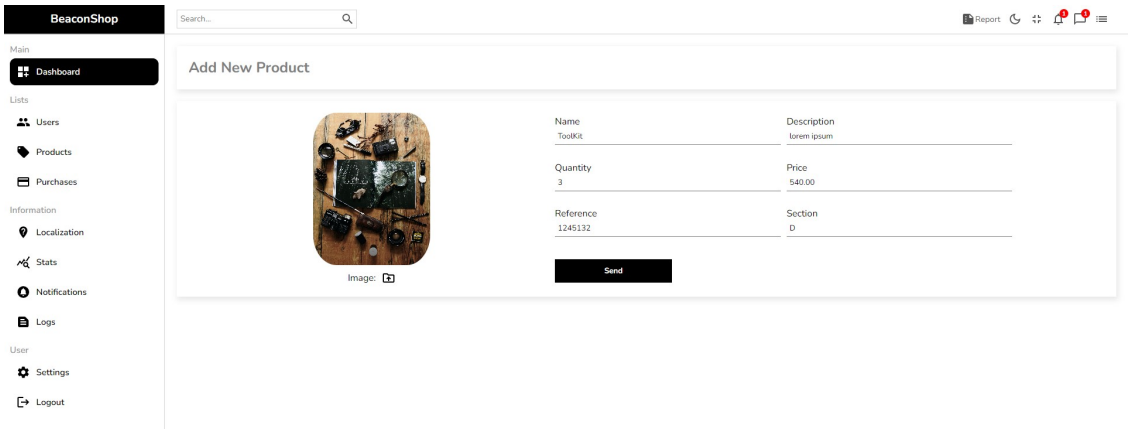


Figure 3.19: New product page.

3.6 Server

The **server** is part of the application tier in the system architecture presented in section 3.3. The server was chosen to be developed with the Express web framework, a framework written in JavaScript and hosted within Node.js. Express helps with the creation of server-side web applications due to its simplicity, flexibility, etc. Since the framework is unopinionated, Express allows us to customize our application to our liking. For the development of the server was used a streamlined code editor, Visual Studio Code, which provides excellent support for JavaScript and has a lighting fast source code editor, ideal for development. Below is listed a small description of each technology used in the server development:

- **Visual Studio Code**[75] - A powerful source code editor which runs on desktop, available for Windows, macOS and Linux. Comes with built-in support for JavaScript, TypeScript and Node.js. The editor includes a rich ecosystem of extensions that provide languages, debuggers, and tools to support the development workflow.

- **JavaScript**[28] - Lightweight, interpreted compiled programming language with first-class functions. JavaScript is a very popular scripting language used by browser environments and non-browser environments like Node.js and Adobe Acrobat. The programming language is considered to be single-threaded, prototype-based, dynamic and multi-paradigm which supports object-oriented, declarative and imperative programming styles.

- **Node.js**[50] - An open-source and cross-platform JavaScript runtime environment which runs on the V8 JavaScript engine that executes code outside of the browser. Node.js is designed to build scalable network applications as an asynchronous even-driven JavaScript runtime.

- **Express**[14] - Unopionated, minimal and flexible Node.js web application framework with a set of features for web and mobile applications. Express is open-source software designed for building web applications and APIs.

3.6.1 Application Modules

In Node.js, Modules are blocks of code in single files or in a collection of multiple files/folders that communicate with an external application based on their functionality, in this case with the developing Express application. Modules are reusable and are used to break down the complexity of code into independent files. There are three types of modules, the Core Modules which come with Node.js installation, the Local Modules, which are built locally in the Node.js application and the Third-party Modules which are modules that are available online (open-source) and are installed in the project folder. The Express application in development uses different types of modules to provide the application with a wide range of functionalities. Below is listed the most relevant Third-party and Local Modules used in this application:

- **BCrypt**[31] - Provides a number of functions to hash and verify passwords.

- **Express**[15] - Fast, unopionionated, minimalist web framework.

- **Http-errors**[23] - Contains methods to create HTTP errors.

- **Jsonwebtoken**[2] - Provides symmetric and asymmetric JSON Web Token implementation.

- **KalmanJS**[6] - Small JavaScript library for using 1D data Kalman filtering.

- **Mongoose**[45] - MongoDB object modelling tool that works in an asynchronous environment.

- **Multer**[46] - Middleware for handling multipart/form-data, which is primarily used for uploading files.

- **Order-id**[12] - Contains methods to generate unique order ids.
- **Trilateration**[20] - Provides a set of functions to perform trilateration calculation.
- **RssiToDistance** - Converts the rssi value to distance.
- **PDFKit**[54] - Javascript PDF generation library for Node and the browser.

3.6.2 REST API

The Express application is going to follow the REST architecture and provide an interface for the client tier to enable communication between a client and the server. This section describes how to implement the REST API for the Express application.

3.6.2.1 Implementation

The HTTP methods are the REST operations and there are four main HTTP methods also known as CRUD operations: GET, POST, PUT and DELETE. In the Express application, we can process these operations by using routes. Routes are defined with the method(path,hadler) and they forward the supported requests to the appropriate function - handler. Each route/HTTP method performs operations on a specific URI which is a REST resource and which also corresponds to the path parameter in the function mentioned. However, each route needs to be defined in the main file of the application and defining routes for each URI is not very efficient as the routes will continually pile up in the main file. Express provides a solution for this with Express. Router objects. Router objects allow the application to create routes for all the URIs to support the client tier on separate files and link them to the main file of the application, leaving one Router per file. This makes the development of the REST API more efficient, organized and flexible. Tables 3.14, 3.15, 3.16, 3.17 display most relevant routes from the the REST API implemented.

Table 3.14: Router for users.

Method	URI	Description
GET	/	Retrieval of all the users from the database.
POST	/	Creation of a user in the database.
POST	/login	Authentication of user credentials.

Table 3.15: Router for beacons.

Method	URI	Description
POST	/	Position Calculation and section retrieval.
GET	/username/:username	Retrieval of the specified user position.
DELETE	/username/:username	Reset of the kalman filter values for the specified user.

Table 3.16: Router for purchases.

Method	URI	Description
POST	/	Creation of a purchase.
GET	/users/:username	Retrieval of purchases made by specified username.
GET	/today	Retrieval of purchases made today.
GET	/receipt/:number	Receipt generation for the purchase with specified number.

Table 3.17: Router for products.

Method	URI	Description
POST	/	Creation of a product.
GET	/sections/:section	Retrieval of products from specified section.
POST	/updateQuality	Increase of product quantity,
GET	/reference/:reference	Retrieval of product with specified reference.

3.6.3 Functionalities

Section 3.6.1 presented the application modules used in the Express application. These modules provide the application with a vast extension of functionalities and allow the application to cover the requirements described in section 3.2. In the following sections will be presented the different and most relevant features of the Express application as well as explain how the modules assisted in their development. Table 3.18 displays most relevant functionalities implemented in the server application.

Table 3.18: Server functionalities.

Section	Functionality
3.6.3.1	Authentication & Authorization
3.6.3.2	Database Connection & Data Modelling
3.6.3.3	Localization Calculation
3.6.3.4	Image Upload
3.6.3.5	PDF Generation

3.6.3.1 Authentication & Authorization

Authentication and authorization are two of the features implemented in the Express application to enforce security. These features are especially used when the client tier is trying to get access to the system. There are also differences between gaining access to the system and gaining access to resources inside the system.

Authentication
Authentication is the process of verifying the identity of a user by processing the credentials and confirming the user's identity with the provided credentials. In case the credentials are valid, the authorization process can then commence. The authorization always succeeds the authentication.

Authorization
Authorization is the process of allowing authenticated users to access the resources inside the system after evaluating their access permissions. By defining the permissions of an authenticated user, the Express application can control the access privileges of the users. Thus, after authentication in the Express application, the user is granted access to resources such as files, information, etc. Authorization delineates the limitations of the users in accessing the system.

Implementation
For implementing authentication and authorization, the Express application has to implement the register and login functionalities. The application implements two routes where the JSON Web Token is used to sign the credentials and bycrypt module to encrypt the password before storing the information in the database. The application also implements a middleware for authentication by creating a route that will require a user token in the header which is the JWT token that is generated in the registration and login routes.

Registration
For the registration functionality, the Express application receives the user input for completing the user details, checks if the details are valid, certify if there is already an existing user, encrypts the user password, creates a user in the database and then at last signs the JWT token. The JWT token signs a payload which in this case will be the username of the user with a private key only known to the Express application.

Login
For the login functionality, the Express application receives the user input, validates the user details, certifies if there is an existing user, verifies with the bcrypt module if the user password corresponds to the password of the user saved in the database and lastly, creates a signed JWT token that is sent as a response. As in the registration functionality, the JWT token signs a payload which in this case will be the username of the user with a private key only known to the Express application.

Authentication Middleware
In the authentication middleware, the Express application checks the existence of the

"x-access-token"header in the request, validates the token, then decodes the token received and certifies if there is an existing user with the username decoded in the token. Regarding authorization, the Express application can pass a boolean value to the middleware specifying if the resource the user is trying to access needs an admin status, if the user doesn't have the permission necessary, the request will be denied. Also, a username header is added to the request for identification purposes, the username is got from the parsed token.

3.6.3.2 Database Connection & Data Modelling

Database connection and data modelling are two features implemented in the Express application that enable the communication between the server and the database tier as well as enforce document data structure via the application layer.

Database Connection
The database connection is an important aspect of the application. By connecting the application to the database, the Express application can store, maintain or access any sort of data of the system.

Data Modeling
Data modelling is an important concept in software development which is responsible for structuring system data with abstract models. These models allow further development of conceptual models and allow the establishment of relationships between data collections. By implementing data models we optimize the quality of system data and reduce the complexity of the system facilitating the querying and management of data in the database.

Implementation
For creating a connection to the database and implementing data models, the Express application uses the Mongoose Module. Mongoose uses a special method for connecting to MongoDB databases and the Models constructors to create the intended data models.

Connection
Mongoose Module facilitates the connection to the MongoDB database. The Module provides the mongoose.connect() function which accepts a simple string as a parameter. The connection can be established to a MongoDB database operating on the local environment or to a MongoDB database operating on a cloud service.

Data Models
To create data modules, Mongoose uses a Model object which is a wrapper on the Mongoose Schema object. The Schema object defines the structure of the document, default values, etc, whereas the Model object provides an interface to the database for creating,

deleting, querying and updating documents. Firstly a Schema is created where document properties are defined through an object where the keys correspond to the properties name in the collection and the values are Schema Types. Schema types can take values of Array, Boolean, Number, etc. Finally, the Model object constructor is called and passed the name of the collection and the defined Schema.

3.6.3.3 Localization Calculation

Localization calculation is a feature implemented in the Express application which calculates the position of a user inside the store. This feature is used when users send the beacon values to the server to update their location and receive the update of the section of the store they are in. This is one of the most important and main features of the application, essential in the system to make the smart shop functional.

Implementation

For the localization calculation, was used three different Modules, one Module which was built locally to be implemented in the application, the RssiToDistance Module, and two Modules installed in the Express application, the KalmanJS Module and the Trilateration Module. Although the names of the modules are self-explanatory, RssiToDistance is a Module which provides a function to convert the RSSI values from the beacons received to distance. KalmanJs is a Module which provides the filtering algorithm called Kalman filter to be applied to the beacon signals to minimize the multipath effects suffered by the signals inside the store. The Trilateration Module provides the localization algorithm called trilateration algorithm which is used to calculate the user position with the received beacon signals values from the request. By applying these three algorithms sequentially, the application is able to calculate the user localization. The Trilateration Module was expanded since the application needed an extra method to process the beacon values.

Location

To calculate the location, the Express application firstly receives the beacon RSSI values from the user and each beacon RSSI value received is processed with a set of methods from different objects before the application calculates the location. Initially, the userBeaconToKalman hashmap is accessed. The hashmap stores the KalmanFilter object from the KalmanJS Module for each username-beacon key. By accessing the hashmap, the application checks if the hashmap has already an existing KalmanFilter object value for the corresponding username-beacon key. If the hashmap returns nothing, a new KalmanFilter object is created. The object then filters the RSSI value from the beacon using the filter() method and the KalmanFilter Object is added to the hashmap with the corresponding username-beacon key. In case there is already an existing KalmanFilter object for the corresponding username-beacon key, the application accesses the object in the hashmap, filters the RSSI value from the beacon with the object's method mentioned

and the hashmap is updated with the new modified KalmanFilter object. Subsequently, the filtered beacon RSSI value is converted to distance using the RssiToDistance Module. All the beacon distances calculated are then set inside an object with the corresponding beacon position coordinates. Since the trilateration algorithm only accepts three distance beacon values, the application only uses the three closest distances. Finally, the Trilateration Module processes the array with three beacon distances and calculates the position of the user. A slight annotation, when a user opens the smartphone application, the application sends a request to the Express application to reset the KalmanFilter objects with the corresponding keys containing the user's username in the userBeaconToKalman hashmap. This is done due to the KalmanFilter objects possibly containing past values filtered while the user is inside the store at a different time or day. The filtering will reset and not consider values from past calculations. Listings 3 and 4 showcase a partial part of the code related to location calculation.

3.6.3.4 Image Upload

Image upload is a feature implemented in the Express application which allows the upload of an image file to the application. This feature is used when a user with admin status sends a request to the server to upload an image through the dashboard of the system. By being able to upload images to the application, new product addition is more practical and convenient, reducing the complexity of expanding the product availability in the store. The product images added to the application will be provided for the smartphone application as well as for the dashboard.

Implementation

For the image upload was used the Multer Module, which is installed in the Express application. Multer is a middleware that is used for handling multipart/form-data, primarily used for uploading files into a system. Multer provides the application with all the necessary functionalities needed to implement the image upload feature.

Upload

For uploading files, the application firstly uses the Multer Module to specify where the application wants to store the files. The application uses the disk storage engine which allows the storing of files on disk. The disk storage engine provides two options, the destination and the filename. The destination is used to specify the destination path for uploaded files. The filename is used to specify the name of the file of the uploaded file. With the disk storage engine setup, the application is ready for the upload of files. When the Express application receives a request with a file attached, the application uses the middleware and the single() method from the Multer Module to accept a single file with the name specified in the method's parameter. The single file is then stored in the application and ready to be fetched by the client tier.

```
//...
const rssiToDistance = require('../rssi_to_distance');
const trilateration = require('../trilateration');
var userBeaconToKalman = new Map();
var beaconPosition = new Map();
//...
beaconPosition.set('1231', {x: 1.90, y:1.68, z: 3.15});
//...
beaconsRouter.use(bodyParser.json());
beaconsRouter.route('/')
.post((req, res, next) => {

    var beaconArray = req.body;
    var username = req.headers["username"];

    var beacons = [];

    for (let beacon of beaconArray)
    {
    console.log(beacon.minor);

    var beaconMajorMinor = beacon.major.toString() + beacon.minor.toString();

    var mapKey = username + ":" + beaconMajorMinor;

    var kf = !userBeaconToKalman.has(mapKey) ?
    new KalmanFilter({R: 0.01, Q: 20, A: 1.22}) : userBeaconToKalman.get(mapKey);

    var newRssi = kf.filter(beacon.rssi);

    var distance = rssiToDistance(newRssi, -61, 2);

    userBeaconToKalman.set(mapKey, kf);

    beacons.push({
        x: beaconPosition.get(beaconMajorMinor).x,
        y: beaconPosition.get(beaconMajorMinor).y,
        z: beaconPosition.get(beaconMajorMinor).z,
        r: distance});
    }
    //...
```

Listing 3: Server Application - Location Calculation - Part 1

3.6.3.5 PDF Generation

PDF generation is a feature implemented in the Express application which generates complex PDF documents dynamically and automatically with customizable data. This

```
    //...
    while(beacons.length > 3)
    {
    var worstBeacon = beacons.reduce(function(prev, curr) {
      return prev.r <= curr.r ? curr : prev;
    });

    beacons.splice(beacons.indexOf(worstBeacon), 1);
    }

    var position = trilateration(beacons[0], beacons[1], beacons[2], true);

    userPosition.set(username, position);

    position["username"] = req.headers["username"];

    Position.create(position)
    .then((PositionRecorded) => {
    console.log('Creating position...');
    console.log('Position Created', PositionRecorded);
    }, (err) => next(err))
    .catch((err) => next(err));
    //...
    res.statusCode = 200;
    res.setHeader('Content-Type', 'application/json');
    res.json(userSection);
})
```

Listing 4: Server Application - Location Calculation - Part 2

feature generates PDF documents for dashboard reports and customer receipts. This is a complementary feature in the system but an essential one in a retail shopping system.

Receipts

Receipts are documents which provide information to customers. These documents include purchase information: date of purchase, number of items purchased, the total price of the purchase, name and location of the business, etc.

Reports

Reports are documents which provide information for the retailers. These documents summarize business sales activities by providing an overview of the sales for a certain period of time. The overview may contain the number of sales, number of products sold, revenue for a given period, etc.

Implementation

For the generation of PDF documents, was used the PDFKit Library, installed in the Express application. PDFkit allows the creation of complex, multi-page, printable documents by providing different features to style and place text, images, etc. The application also uses the fs module from Node.js to interact with the file system.

Document Creation

For creating the custom PDF documents, the application has to firstly create a new document by creating a PDFDocument object. To customize the document, the application uses different methods provided by the library. An example is the text which uses the font(), fontSize() and text() methods. For certain methods like text() we specify the position of x and y to decide the placement of the PDF components. When the customization of the PDF document is finished, the application uses the end() to finalize the PDF file and calls the createWriteStream() method from the fs module to save the PDF document with a specified document name and file path.

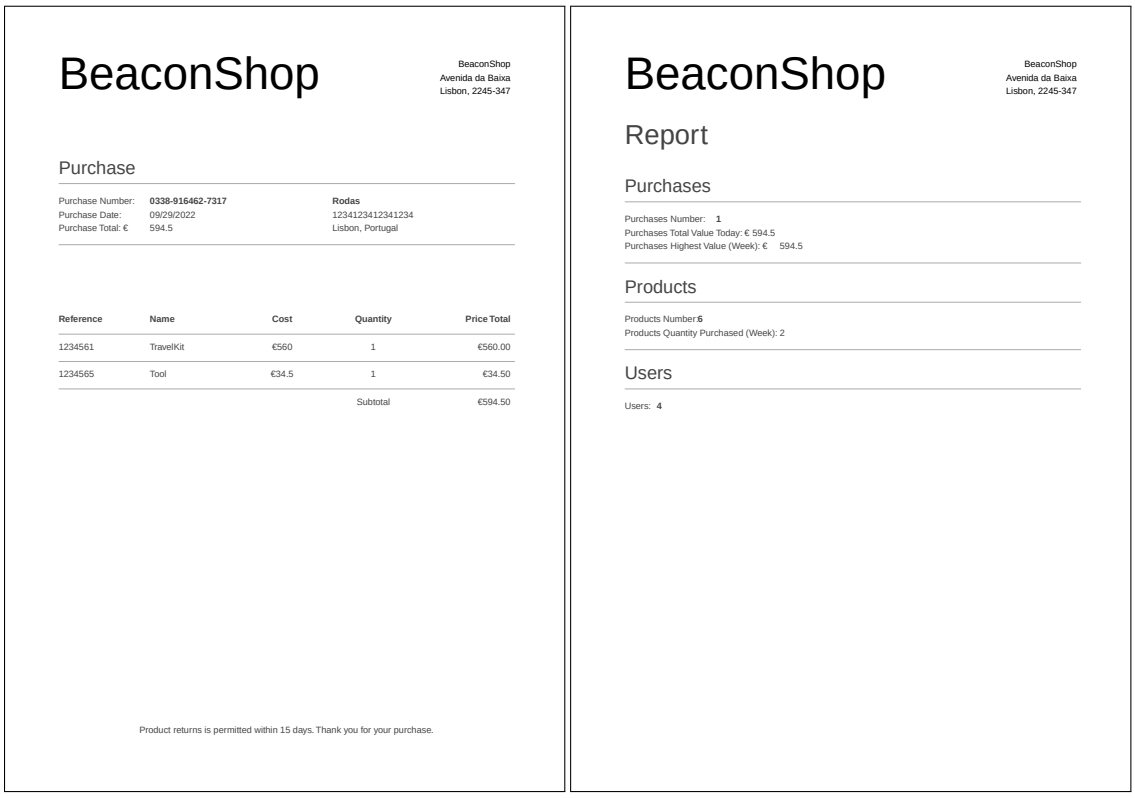


Figure 3.20: A generated PDF document - receipt and report.

3.7 Database

The **database** is part of the data tier in the system architecture presented in section 3.3. The type of database chosen to be implemented in the system was the MongoDB [44]

database, a document-oriented NoSQL database used for large-scale data storage. MongoDB allows the storage of structured and unstructured data as well as the storage of documents in JSON-like format, a format commonly used in most modern programming languages. The server, smartphone application and the web application from the system implemented commonly process requests with JSON objects present in the payload of the request thus it is logically more efficient to keep the format of the data normalized throughout the whole system. Regarding the implementation, the database was implemented with MongoDB Atlas [43], a fully-managed cloud database that handles the complexity of deployment and management on a cloud service provider. MongoDB Atlas covers the needs for infrastructure as well as provides real analytics and rich visualizations of the database.

3.7.1 Data Structure & Relationships

Data structuring is an important feature of the system which allows easy querying and manipulation of the data in the database. The data structure is forced on the application layer of the server tier by implementing data models to support the system needs - feature described in section 3.6.3. In the next section is described how to establish the relationships between the different data models.

3.7.1.1 Relationships

Although MongoDB is a non-relational document database, there is a solution to create relationships between the collections of JSON documents. There are two ways of ensuring the relationships - through embedded documents and references. While embedded documents are documents nested inside the data of other documents which provide a distinct notion of relationship, references are plainly logical, something that needs to be documented to be understood. Both ways of establishing relationships are important and implement different types of relationships between data [41].

Embedded Documents

Embedded documents implement One-to-One and One-to-Many relationships by embedding a document or an array of documents inside another document. These relationships are defined directly in the data and can be instantly observed. Figure 3.21 shows an example model from the system that contains embedded documents relationship. The relationships are listed below:

- **One-to-One** - Basic embedded document provides One-to-One relationship. One parent document is related to one child document.
- **One-to-Many** - Implementation of the One-to-Many relationship requires more complexity and is implemented by embedding an array of documents inside a document. One parent document is related to multiple child documents.

```
const purchaseSchema = new Schema({
    number: {
        type: String,
        required: true,
        unique: true
    },
    username: {
        type: String,
        required: true,
    },
    date: {
        type: Date,
        required: true
    },
    card: Card.schema,
    products: [Product.schema],
    total: {
        type: Number
    }
});
```

Figure 3.21: Purchase model with one-to-one (Card) and one-to-many relationships (Products). A purchase can only have one card associated with many products.

Document References

References implement Many-to-One and Many-to-Many relationships by referencing a field from one document inside another document. These relationships are hard to observe in the data models thus it is important to document these types of relationships. Figure 3.22 shows an example model from the system that contains document references relationship. The relationships are listed below:

- **Many-to-One** - An unlimited number of documents contain a defined field which references one unique document. The field can be an id field or a unique field from a document.
- **Many-to-Many** - Implementation of the Many-to-Many relationship requires advanced referencing and is implemented by a document defining a field of an array of references of other documents. One parent document is related to multiple child documents. The field can contain an array of id fields or unique fields from a document.

3.7.2 Data Diagram

Having presented all types of relationships and data structuring conceivable in MongoDB with the implementation of data modelling, it is now possible to build a database diagram which displays all the structure and relationships of the database in the system developed. Figure 3.23 shows the database diagram.

```
const purchaseSchema = new Schema({
    number: {
        type: String,
        required: true,
        unique: true
    },
    username: {
        type: String,
        required: true,
    },
    date: {
        type: Date,
        required: true
    },
    card: Card.schema,
    products: [Product.schema],
    total: {
        type: Number
    }
});
```

```
var UserSchema = new Schema({
    username: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    token: {
        type: String
    },
    admin: {
        type: Boolean,
        default: false
    },
    active: {
        type: Boolean,
        default: false
    }
});
```

Figure 3.22: Purchase model with many-to-one relationship (referencing unique username field). Many purchases can be related to solely one user.

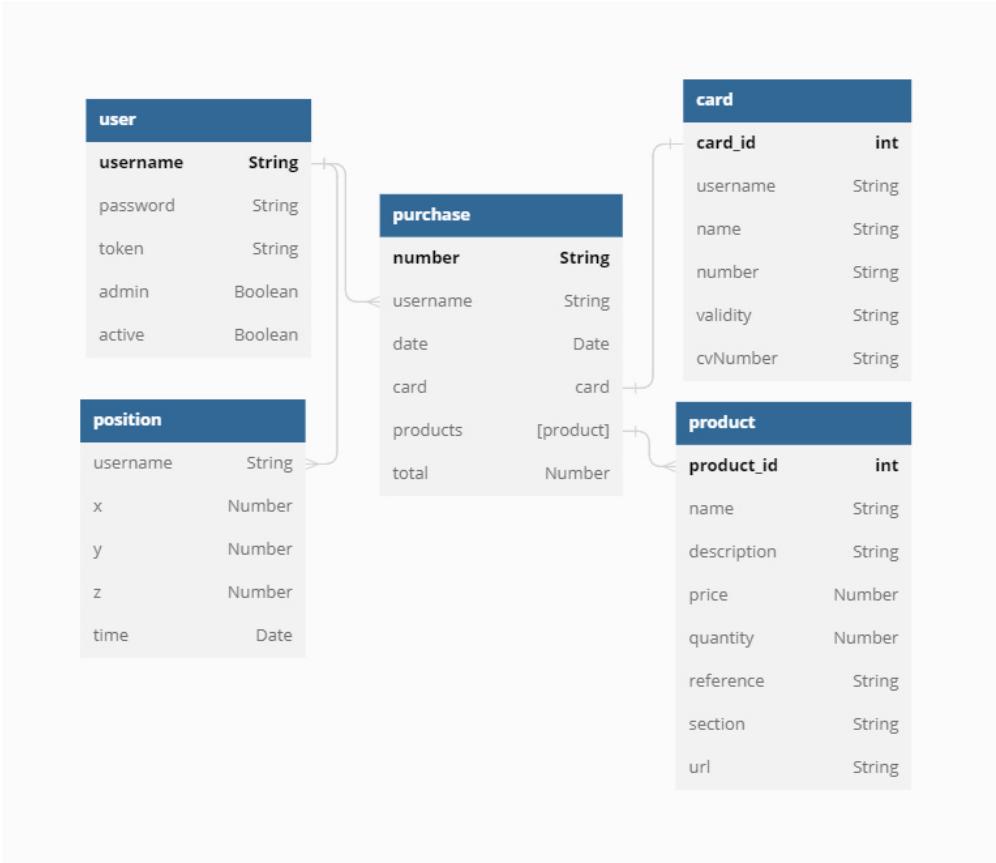


Figure 3.23: Database Diagram

4

System Evaluation & Testing

The previous chapter described an approach taken in the implementation of the system capable of satisfying the needs of the requirements proposed. In this chapter, the system localization techniques implemented are evaluated with quantitative and qualitative methodologies. The quantitative evaluation provides statistical assessments that produce data and outcomes of different numeric measurements. Qualitative evaluation is more abstract and provides valuable insight into whether the system achieves the intended objectives. Multiple functionalities of the system of both web and smartphone applications are also tested. Each application runs through different scenarios to analyze and verify if the system fulfils the requirements presented to the system.

4.1 System Beacons

The system is based on the use of beacon technology, one of the main parts of the system. In the system evaluation and testing, the Ultra BLE beacons from blueup were used, show in figure 4.1. These beacons were designed for indoor BLE implementations and have full support for the iBeacon protocol which was used in the development of the system. Due to its directive antennas with circularly-polarized polarization, we were allowed to have improved accuracy with reduced multi-path effects from the environment [5].

Figure 4.1: Ultra BLE beacons from blueup [5].

4.2 Evaluation

The evaluation is necessary to determine the accuracy of the system in locating the costumers inside the retail store as the whole system depends on the outcome of the localization techniques deployed in the system. To display the products in the retail store the approach taken was to divide the room into different sections where each section includes different products. When the system determines in which section the user is located, the system proceeds to send the corresponding products of the section. For the evaluation of the system, real-life experiments were conducted in an empty room without obstacles of 6 by 5 meters with 3.15 meters in height where four beacons were installed on the ceiling of the room at known locations, as shown in figure 4.3. For the experiments, users walked through a predefined path with the smartphone in their hands. The smartphone used is an iPhone 8 Plus. Throughout the predefined path were placed several markers at which upon the arrival of the user, a button on a smartphone application was pressed by the user to send to the server the beacon signals received. The server would then calculate and store the location coordinates, which were afterwards used for the evaluation. Ground truth was collected by making separate measurements of the path defined and the markers. The distance between the estimated location calculated by the localization techniques and the ground truth results in the instantaneous localization error.

4.2.1 Quantitative Evaluation

Trilateration and Kalman Filtering are the algorithms implemented in the system to compute the localization of the clients inside the retail store. Trilateration determines the location of the client with distance information between the client and three reference points which are the location of the installed beacons. Since the system has to deal with RSSI values which are prone to suffer from multi-path effects, Kalman Filtering is used to reduce the error that may occur with the RSSI values. To see the effectiveness of the localization techniques implemented in the system, the experiments were evaluated with and without Kalman filtering separately and recorded the results. Figure 4.2 shows the performance of trilateration estimation error without and with Kalman filtering. The figure shows that the distance estimation performed by the system shows a higher performance when Kalman filtering is applied, the filtering thus indeed improves accuracy by some margin as using Kalman demonstrates decreasing the error caused by multi-path effects the signals suffer in the environment. Comparing the results of the distance estimation obtained by both approaches evaluated, the trilateration distance estimation with Kalman filtering, with a mean distance error of ≈ 0.51 meters, proved to be 28.32% more efficient than without using the Kalman filtering, with a mean distance error of ≈ 0.72 meters. In conclusion of the quantitative outputs, a better result can be possibly achieved by modifying the parameters used in the conversion of the RSSI values to distance although a more in-depth analysis of the environment might be needed by creating a path-loss

model for the beacons. Considering the size of the room, the decision of installing 4 beacons can possibly be a factor in decreasing the performance of the localization techniques through signal interference. Also to measure the effectiveness of the system in a retail shop environment, these experiments should be also evaluated in a real-life scenario, using a common retail store as an experimental location. There may have been some errors in calculations of the ground truth because of human error since there were employed manual techniques to calculate the different localization points in the room. However, the localization techniques implemented produced positive results, enough to make for a minimum viable product.

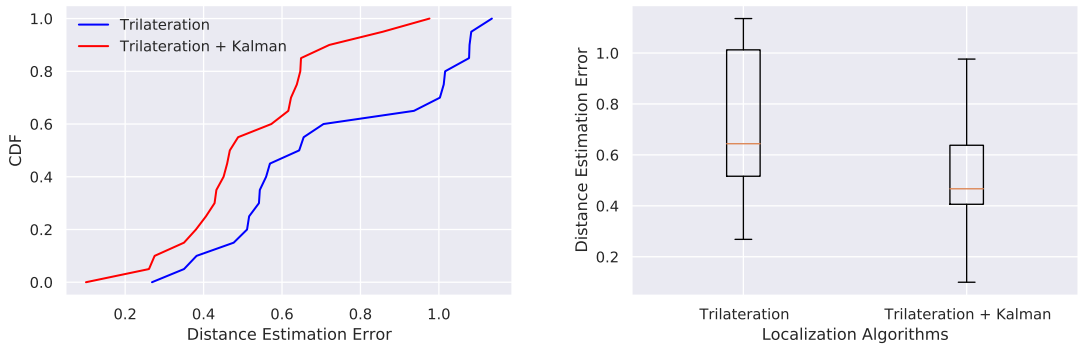


Figure 4.2: Distance estimation error using Trilateration and Trilateration + Kalman.

4.2.2 Qualitative Evaluation

Figure 4.3 shows the predefined path tagged with markers, the positions where beacons were installed in the room and one of the resulting paths from the experiments conducted with the calculated positions by the localization techniques implemented in the system. In comparison to the real positions, although the figure shows the estimated positions with some deviations, the localization techniques presented a satisfactory result in computing the user path. For the display of products inside the retail store, the system divides the store into different sections as described in the evaluation context. By examining figure 4.3, the best approach to take is to not define small area sections since the estimated positions can suffer some deviations, causing the system to imprecisely display the wrong products to the user. This issue can be possibly mitigated if the system implements the new changes and evaluations of different scenarios addressed in the quantitative section for future work. Verified by the quantitative evaluation, from the qualitative evaluation observations we can conclude that the localization techniques implemented in the system can estimate a decent user path enough to provide the system with performance making for a minimum viable product.

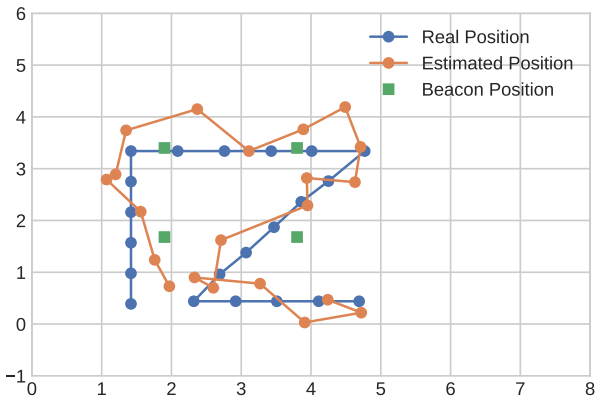


Figure 4.3: Path computed by the system using Trilateration + Kalman.

4.3 Testing

The testing process conducted focused on the performance of the system as a whole. Specifically, the testing was performed on the smartphone and web application, client-system interactive components of the system. For each part of the testing process carried out, different scenarios were performed to determine whether the system achieved the requirements proposed in the implementation of the system. The components were also tested to see in how much clicks or taps the scenarios were performed. By checking these requirements, the system can be considered a minimum viable product with enough features to be usable and validated in the system development cycle. Hence, the next sections present different testing scenarios to check if the system covers each objective.

4.3.1 Smartphone Application

The smartphone application is part of the client tier in the system. The main objective of the application is to connect the clients of the retail store to the system and display to them the available products in the retail store, provide product information, process payments for the products, display purchase information, etc. Below are presented different scenarios tested on the smartphone application.

4.3.1.1 Scenario 1 - Authentication and Registration

These two functionalities aim to provide security to the system and ensure client privacy. The application conveys the resources of the system only in the event that the client has authenticated. The goal is to also provide user identification so that the user is able to request the system to deliver personal information of client activity in the retail store securely. This objective was fulfilled, the authentication and registration processes in the

user interface are quick and user-friendly, allowing clients to perform these functionalities easily. The registration process can be carried out within 5 taps on the screen, without considering the taping of text input. The authentication process can be carried out within 3 taps on the screen also without considering the taping of the text input. If the user has been previously authenticated, the system ensures automatic authentication which improves the user experience, going through the authentication with 0 taps.

4.3.1.2 Scenario 2 - Product Search

Product display is an integral part of the application, as the products are displayed in a dynamic grid, the application provides a simple efficient approach to visualize multiple products of the retail store. The products are updated on the user interface as the client roams through the retail store, entering or changing sections of the store predefined in the system. The product cards are fast responsive and the display of single product information is well organized, presenting clear details about the product and an intuitive addition button to store the products in the cart system of the application. This objective was fulfilled as a user can accomplish the search and addition of products of their interest. The addition of products can be done within 1 or 3 taps, depending on whether the user enters the view with single product information displayed.

4.3.1.3 Scenario 4 - Products Payment

Essential feature of an automatic retail store which offers simplicity for clients' shopping experience. Avoiding huge lines of checkout, the client is able to make different and several purchases in no means time by only using their smartphone. This objective is fulfilled with the automated and simple payment feature implemented in the application which provides payment flexibility and more security. The payment can be done within 6 taps without considering the taping of inputs.

4.3.1.4 Scenario 5 - Purchases Information Access

Purchase history is convenient, enabling access to information about previous purchases performed by the client. In an automatic retail store, this implementation is essential and crucial as a paper receipt replacement and brings convenience in delivering purchase receipts on demand in a practical manner. This objective is fulfilled as the client can search through past purchases and see their detailed information. The client can also request a PDF document type of receipt with purchase information that can be stored locally on the smartphone. The purchase access information can be done within 2 to 3 taps.

4.3.1.5 User Experience

In this section, an iPhone with the smartphone application developed was handed over to users for experimental purposes. In this experiment, 26 users with a range of age between 20 and 51 were instructed to use the application in a shopping activity scenario and recreate the scenarios described in the previous sections. The users were tasked to rate the application from values of 0-10, being value 0 as the lowest rating - very negative - and value 10 as the highest rating - very positive. The following questions were asked after the recreated scenarios, "*How easy to use do you consider the smartphone application?*", "*How fast functioning do you consider the smartphone application?*", and "*How attractive do you find the user interface of the smartphone application?*". From these results we were able to obtain an idea of how efficient, attractive and intuitive the smartphone application is, yielding the results shown on graphs in figure 4.4.

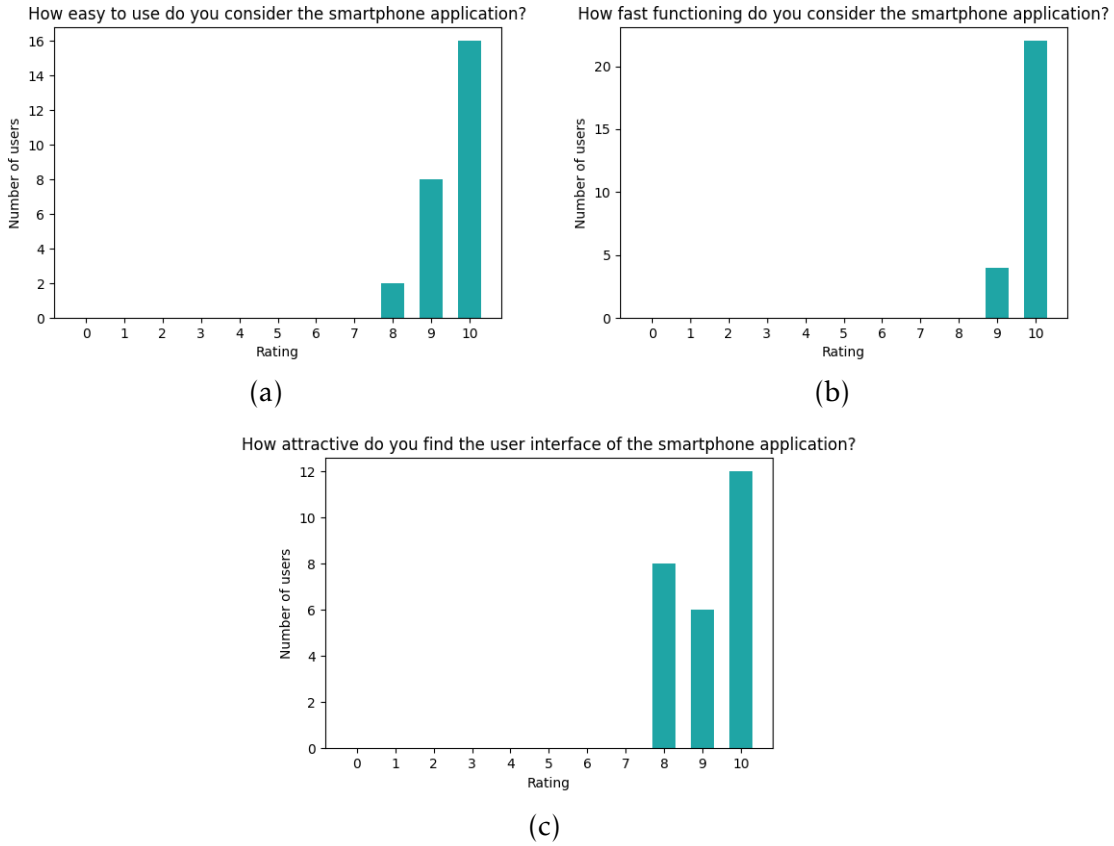


Figure 4.4: User experience on different shopping activity scenarios using the smartphone application.

4.3.2 Web Application

The web application also defined as the dashboard is part of the client tier in the system. The main objective of the application is to provide the admin or staff of the retail store

with useful information about the store's state and present events. The dashboard allows the admin to set daily goals of revenue, track recent purchases, check user or product information, etc. Below are presented different scenarios tested on the web application.

4.3.2.1 Scenario 1 - Authentication

Authentication on the web application has similar functionality to the authentication on the smartphone application although the authorization level needed for access is unlike. In the web application, authentication is used to ensure access to only admin or staff workers of the retail store as it provides security to valuable business information which must be private and only accessed by trusted users. Successful authentication in the web application allows the user to request the system to deliver and display store private information of clients and store activity. This objective was fulfilled, admins can authenticate in the application through the user interface developed and if the user has been previously authenticated, the system ensures automatic authentication which improves the operation of the admin on the dashboard. The authentication process can be carried out within 3 clicks on the screen also without considering the clicking or taping of the text input. As the system ensures automatic authentication, authentication can be gone through with 0 clicks.

4.3.2.2 Scenario 2 - Dashboard Visualisation

The main page of the web application holds the highest importance to the dashboard. Dashboards are effective tools for monitoring and visualizing data owing to their innate visual nature and capacity to quickly gather and organize information. This objective was fulfilled, the dashboard allows the admin to supervise store events and different metrics regarding the store state in real-time as the main and most important information is aggregated on one page. To visualize main information on the dashboard 0 clicks are needed.

4.3.2.3 Scenario 3 - Products/Purchases/Users Search

The web application has several distinct essential pages implemented. The admin should contain the option to have more detailed information about a purchase, product, or user, therefore the dashboard has to be able to supply the admin with more in-depth details. This objective was fulfilled, the admin can open listing pages of products, purchases, users and follow through by opening a single page with the respective information about the searched resource and see relevant data on different visualisation components. The access to a product, purchase or user can be achieved in 2-3 clicks.

4.3.2.4 Scenario 4 - Product Creation

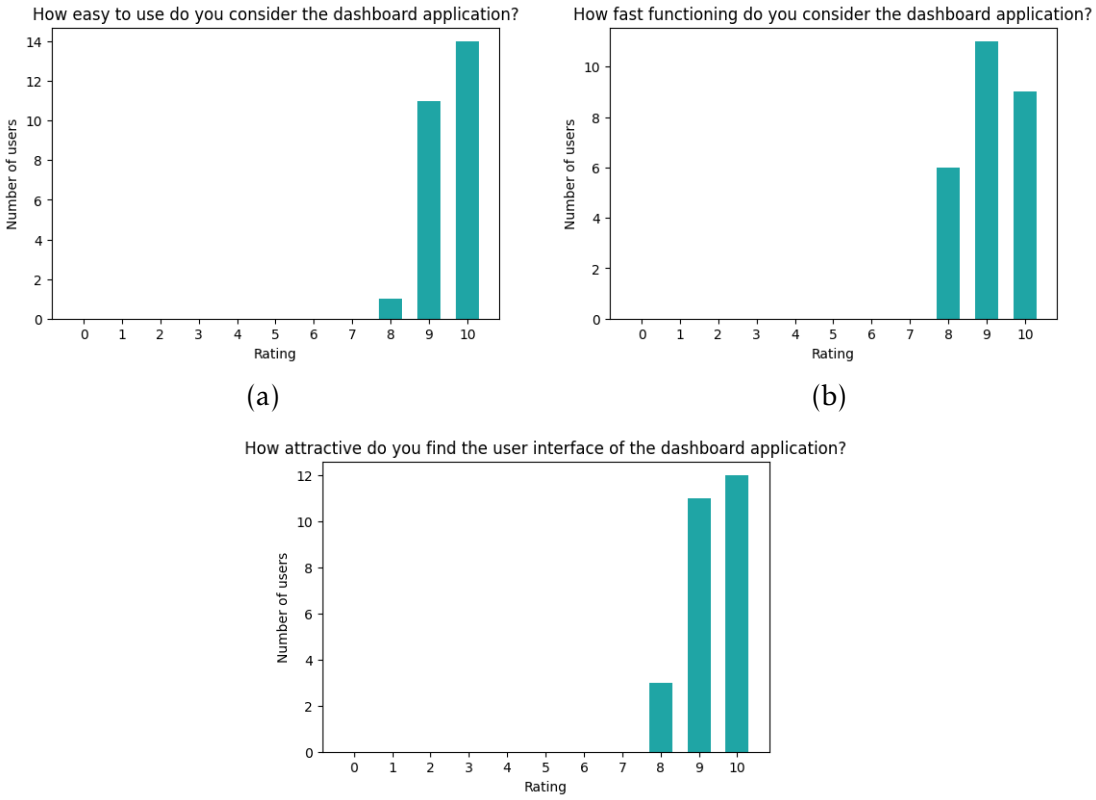
Retail stores have to ensure that their business is meeting customer's demands an uphold product stock. The clients' shopping is highly dependant on the system of the retail store. Admin or staff workers should be able to manage stores' product stock by updating with creation of new products or updating existing ones with new product information. These products are meant to be displayed on the smartphone application where the shopping logic is performed. This objective was fulfilled, the admin can submit new products with image upload through the product creation page on the dashboard. The product creation can be achieved within 11 clicks.

4.3.2.5 Scenario 5 - Report

Business report is a significant peculiarity for retail growth. A business report may include inventory reports, product performance, sales summary, etc. Based on this information, retailers have the means for creating customer recommendations based on their purchase history and improve customer shopping experience. This objective was fulfilled, the admin can request a PDF document type of report with store information. The request of the report can be carried out solely with 1 click.

4.3.2.6 User Experience

In this section, the dashboard application developed was presented to users for experimental purposes. In this experiment, 26 users with a range of age between 20 and 51 were instructed to use the application in a monitoring scenario of store activity and recreate the scenarios described in the previous sections. The users were tasked to rate the application from values of 0-10, being value 0 as the lowest rating - very negative - and value 10 as the highest rating - very positive. The following questions were asked after the recreated scenarios, "*How easy to use do you consider the dashboard application?*", "*How fast functioning do you consider the dashboard application?*", and "*How attractive do you find the user interface of the dashboard application?*". From these results we were able to obtain an idea of how efficient, attractive and intuitive the dashboard application is, yielding the results shown on graphs in figure 4.4.



(a) (b) (c)

Figure 4.5: User experience on different shopping activity scenarios using the smartphone application.

5

Conclusion

This chapter highlights the contents addressed throughout this thesis, stating the context of the development of the solution, the several adopted applications and whether the system meets the objectives proposed by the thesis. After summarizing these contents, this chapter presents the main obstacles encountered during the development of the system and closes the chapter with a list of possible improvements or feature additions to be incorporated for future work.

5.1 Conclusions

The aim of this thesis is to present a practical system that addressed the challenges of implementing an automated retail shop with low energy consumption by integrating BLE beacon technology. With the use of BLE beacons, the system calculates the position of the clients in the store and displays the corresponding products based on the location whilst the clients perform their shopping through the use of their smartphones. The developed system ensures the necessary functionalities for a minimum viable product to operate the automated retail store providing different applications that allow the interaction between clients, personnel and the automated store as well as the application responsible for computing all the logic of the localization estimation and processing of the data in the system. Through the evaluation, where several measurements were performed to evaluate the performance of the localization techniques, and the testing conducted on the applications of the system, we generated enough performance results in localization calculation and application testing to confirm that the system is suitable to be implemented in the context of an automated retail store, fulfilling the purpose of this thesis and opening new innovative possibilities for retail automation for further exploration or improvement on the proposed system.

The evaluation and the testing performed on the system confirmed the completion of the several Research Objectives defined in the introduction of the thesis. The tests conducted in our application, precisely in evaluation, have proved a margin of error of 0.51 meters in the estimation of client location which is perfectly acceptable for our

use case, ensuring a good experience for the user during their purchases. Therefore we can conclude that we can count on the reliability of the system in presenting the products to the user. The tests carried out on the mobile phone and web applications were very positive as all the scenarios tested were fulfilled and the functionalities were performed in a few clicks. Meaning that the system applications were able to fulfil the functional and non-functional requirements defined. Customers are presented with a functional platform, a decent mobile phone application to interact with the shop and make purchases or access other types of functionalities proposed by the system. The shop workers also have a good visualization platform, a dashboard, to manage the store or monitor the purchases made by the customers. The dashboard works in real-time where workers are notified of recent purchases. With this, we have a sufficient solution for the automation of a retail shop.

Considering the applications of the developed system in the retail industry, the best fitting retailers would be the home furnishings, clothing, home-improvement and gardening retailers. Taking IKEA as an example, a home furnishing retailer, some of the processes in their stores like the registration of the reference of the products the clients are interested in purchasing, the payment and the shipping processes can be all automated with the use of the smartphone application and dashboard which could be easily customized to attend these functionalities. Implementing the developed system in this kind of retail store would highly diminish the complexity of the logistics of many operations while also improving the client shopping experience. It is questionable whether this system would work in retail stores like supermarkets due to the large number of products confined in small sections to be displayed on the smartphone application. Although there are two solutions which could be implemented to solve this problem, the developed system would only work in these kinds of environments if the localization error would be further minimized to define smaller store sections with products to display or if there would be a creative new way of displaying the products on the user interface, differently from what was implemented, by displaying the products as in shelves for example.

5.2 Limitations & Future Work

This thesis proposes a system to automate a retail store using beacon technology. Despite the system's efficiency, there is still much potential for improvements and further studies about new or already used technologies in the system. Below are discussed the limitations found in the system and suggested possible implementations for future research.

Even though the computed mean error of the localization estimation resulted in a low error value, a better result could be achieved by creating a model of path-loss to make the best fit for the parameters in the method of conversion from RSSI values to distance for the beacons installed. To further increase the accuracy of the localization method, a study on the optimal parameters for the Kalman filtering could be conducted since the preprocessing showed to be a relevant implementation of enhancing accuracy. Another

possible approach to take is to implement and evaluate different localization algorithms, possibly the ones mentioned in the state of the art in this thesis to see which algorithm better fits in the environment of a retail store.

The system's evaluation could take place in a real-world setting, such as a big retail establishment with moving clients during their shopping activities. This would simulate a real-life scenario which could guarantee more genuine results of the performance of the system in a retail store.

A suggestion for new features to be added to the system for automated retail stores is using machine learning and stored customer activity information. Using machine learning it is possible to analyse the data generated by the clients and create personalised recommendations to customers of products of interest or to assist retailers in their marketing strategy decisions. It is also possible, using the data of the path taken by customers within the shop, to create new product placement strategies.

Bibliography

[1] T. Andersson. *Bluetooth low energy and smartphones for proximity-based automatic door locks*. 2014 (cit. on p. 9).

[2] *auth0/node-jsonwebtoken: JsonWebToken implementation for node.js http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html*. URL: https://github.com/auth0/node-jsonwebtoken#readme (visited on 02/05/2023) (cit. on p. 62).

[3] P. Barsocchi, M. Girolami, and D. La Rosa. "Detecting Proximity with Bluetooth Low Energy Beacons for Cultural Heritage". en. In: *Sensors* 21.21 (Oct. 2021), p. 7089. ISSN: 1424-8220. DOI: 10.3390/s21217089 (cit. on p. 8).

[4] R. Bembenik and K. Falcman. "BLE Indoor Positioning System Using RSSI-based Trilateration". en. In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 11.3 (Sept. 2020), pp. 50–69. DOI: 10.22667/JOWUA.2020.09.30.050 (cit. on pp. 17, 18).

[5] *BlueUp*. en. URL: https://www.blueupbeacons.com/index.php?page=ultra (visited on 02/05/2023) (cit. on p. 75).

[6] W. Bulten. *KalmanJS*. original-date: 2015-09-24T21:13:05Z. Sept. 2022. URL: https://github.com/wouterbulten/kalmanjs (visited on 02/05/2023) (cit. on p. 62).

[7] J. H. Choi and H.-J. Lee. "Facets of simplicity for the smartphone interface: A structural model". en. In: *International Journal of Human-Computer Studies* 70.2 (Feb. 2012), pp. 129–142. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2011.09.002 (cit. on p. 25).

[8] *Core Location | Apple Developer Documentation*. URL: https://developer.apple.com/documentation/corelocation (visited on 02/05/2023) (cit. on p. 41).

[9] *CSS: Cascading Style Sheets | MDN*. en-US. URL: https://developer.mozilla.org/pt-BR/docs/Web/CSS (visited on 02/05/2023) (cit. on p. 55).

[10] T. Dag and T. Arsan. "Received signal strength based least squares lateration algorithm for indoor localization". en. In: *Computers & Electrical Engineering* 66 (Feb. 2018), pp. 114–126. ISSN: 00457906. DOI: 10.1016/j.compeleceng.2017.08.014 (cit. on pp. 17, 22).

[11] S. Darroudi and C. Gomez. "Bluetooth Low Energy Mesh Networks: A Survey". en. In: *Sensors* 17.7 (June 2017), p. 1467. ISSN: 1424-8220. DOI: 10.3390/s17071467 (cit. on p. 5).

[12] M. DeRazon. *order-id*. original-date: 2016-11-22T11:18:03Z. Sept. 2022. URL: https://github.com/mderazon/order-id (cit. on p. 63).

[13] *Eddystone*. original-date: 2015-06-26T10:58:02Z. Feb. 2022. URL: https://github.com/google/eddystone (cit. on p. 13).

[14] *Express - Node.js web application framework*. en. URL: https://expressjs.com/ (visited on 02/05/2023) (cit. on p. 62).

[15] *expressjs/express*. original-date: 2009-06-26T18:56:01Z. Sept. 2022. URL: https://github.com/expressjs/express (visited on 02/05/2023) (cit. on p. 62).

[16] R. Faragher and R. Harle. "Location Fingerprinting With Bluetooth Low Energy Beacons". en. In: *IEEE Journal on Selected Areas in Communications* 33.11 (Nov. 2015), pp. 2418–2428. ISSN: 0733-8716. DOI: 10.1109/JSAC.2015.2430281 (cit. on p. 22).

[17] M. Fazio et al. "A proximity-based indoor navigation system tackling the COVID-19 social distancing measures". en. In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. Rennes, France: IEEE, July 2020, pp. 1–6. ISBN: 978-1-72818-086-1. DOI: 10.1109/ISCC50000.2020.9219634 (cit. on p. 8).

[18] *Foundation | Apple Developer Documentation*. URL: https://developer.apple.com/documentation/foundation (visited on 02/05/2023) (cit. on p. 43).

[19] O. I. Franko and T. F. Tirrell. "Smartphone App Use Among Medical Providers in ACGME Training Programs". en. In: *Journal of Medical Systems* 36.5 (Oct. 2012), pp. 3135–3139. ISSN: 0148-5598, 1573-689X. DOI: 10.1007/s10916-011-9798-7 (cit. on p. 24).

[20] G. Heja. *trilateration.js*. original-date: 2015-07-03T20:57:19Z. Sept. 2022. URL: https://github.com/gheja/trilateration.js (cit. on p. 63).

[21] R. Heydon. *Bluetooth low energy: the developer's handbook*. en. Upper Saddle River, NJ Boston Indianapolis [und 13 weitere]: Prentice Hall, 2015. ISBN: 978-0-13-288836-3 (cit. on p. 6).

[22] *HTML: HyperText Markup Language | MDN*. en-US. URL: https://developer.mozilla.org/pt-BR/docs/Web/HTML (visited on 02/05/2023) (cit. on p. 55).

[23] *http-errors*. original-date: 2014-09-08T20:02:20Z. Sept. 2022. URL: https://github.com/jshttp/http-errors (visited on 02/05/2023) (cit. on p. 62).

[24] J.-H. Huh and K. Seo. "An Indoor Location-Based Control System Using Bluetooth Beacons for IoT Systems". en. In: *Sensors* 17.12 (Dec. 2017), p. 2917. ISSN: 1424-8220. DOI: 10.3390/s17122917 (cit. on p. 19).

[25] A. Inc. *Swift - Apple Developer*. en. URL: https://developer.apple.com/swift/ (visited on 02/05/2023) (cit. on p. 40).

[26] A. Inc. *SwiftUI Overview - Xcode - Apple Developer*. en. URL: https://developer.apple.com/xcode/swiftui/ (visited on 02/05/2023) (cit. on p. 40).

[27] A. Inc. *Xcode 14 Overview*. en. URL: https://developer.apple.com/xcode/ (visited on 02/05/2023) (cit. on p. 40).

[28] *JavaScript | MDN*. en-US. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (visited on 02/05/2023) (cit. on pp. 54, 62).

[29] K. E. Jeon et al. "BLE Beacons for Internet of Things Applications: Survey, Challenges, and Opportunities". en. In: *IEEE Internet of Things Journal* 5.2 (Apr. 2018), pp. 811–828. ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2788449 (cit. on pp. 7, 8, 25).

[30] M. Ji et al. "Analysis of positioning accuracy corresponding to the number of BLE beacons in indoor positioning system". en. In: *2015 17th International Conference on Advanced Communication Technology (ICACT)*. Phoenix Park, PyeongChang, South Korea: IEEE, July 2015, pp. 92–95. ISBN: 978-89-968650-5-6. DOI: 10.1109/ICACT.2015.7224764 (cit. on p. 7).

[31] *kelektiv/node.bcrypt.js: bcrypt for NodeJs*. URL: https://github.com/kelektiv/node.bcrypt.js#readme (visited on 02/05/2023) (cit. on p. 62).

[32] T. Kluge, C. Groba, and T. Springer. "Trilateration, Fingerprinting, and Centroid: Taking Indoor Positioning with Bluetooth LE to the Wild". en. In: *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. Cork, Ireland: IEEE, Aug. 2020, pp. 264–272. ISBN: 978-1-72817-374-0. DOI: 10.1109/WoWMoM49955.2020.00054 (cit. on pp. 19, 20).

[33] M. Kohne and J. Sieck. "Location-Based Services with iBeacon Technology". en. In: *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*. Madrid: IEEE, Nov. 2014, pp. 315–321. ISBN: 978-1-4799-7600-3. DOI: 10.1109/AIMS.2014.58 (cit. on pp. 10–12).

[34] K. Komai et al. "Beacon-based multi-person activity monitoring system for day care center". en. In: *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. Sydney, Australia: IEEE, Mar. 2016, pp. 1–6. ISBN: 978-1-5090-1941-0. DOI: 10.1109/PERCOMW.2016.7457140 (cit. on p. 9).

[35] I. V. Kovalev et al. "Model of the reliability analysis of the distributed computer systems with architecture "client-server"". en. In: *IOP Conference Series: Materials Science and Engineering* 70 (Jan. 2015), p. 012009. ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/70/1/012009 (cit. on p. 23).

[36] D. Kunda and H. Phiri. "A comparative study of nosql and relational database". In: *Zambia ICT Journal* 1.1 (2017), pp. 1–4 (cit. on p. 28).

[37] J. Lindh. "Bluetooth low energy beacons". In: *Texas Instruments* (2015), p. 2 (cit. on p. 7).

[38] A. Mackey, P. Spachos, and K. N. Plataniotis. "Enhanced Indoor Navigation System with Beacons and Kalman Filters". en. In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Anaheim, CA, USA: IEEE, Nov. 2018, pp. 947–950. ISBN: 978-1-72811-295-4. DOI: 10.1109/GlobalSIP.2018.8646581 (cit. on p. 15).

[39] L. Mainetti, L. Patrono, and I. Sergi. "A survey on indoor positioning systems". en. In: *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. Split, Croatia: IEEE, Sept. 2014, pp. 111–120. ISBN: 978-953-290-052-1. DOI: 10.1109/SOFTCOM.2014.7039067 (cit. on p. 7).

[40] *Material Icons - Material UI*. URL: https://mui.com/material-ui/material-icons/ (visited on 02/05/2023) (cit. on p. 55).

[41] *Model Relationships Between Documents — MongoDB Manual*. en. URL: https://www.mongodb.com/docs/manual/applications/data-models-relationships/ (visited on 02/05/2023) (cit. on p. 72).

[42] *Moment.js | Home*. URL: https://momentjs.com/ (visited on 02/05/2023) (cit. on p. 55).

[43] *MongoDB Atlas Database | Multi-Cloud Database Service*. en-us. URL: https://www.mongodb.com/atlas/database (visited on 02/05/2023) (cit. on p. 72).

[44] *MongoDB: The Developer Data Platform*. en-us. URL: https://www.mongodb.com (visited on 02/05/2023) (cit. on p. 71).

[45] *Mongoose ODM v6.6.2*. URL: https://mongoosejs.com/ (visited on 02/05/2023) (cit. on p. 62).

[46] *Multer*. original-date: 2014-01-30T18:16:04Z. Sept. 2022. URL: https://github.com/expressjs/multer (cit. on p. 62).

[47] S. Mumbaikar, P. Padiya, et al. "Web services based on soap and rest principles". In: *International Journal of Scientific and Research Publications* 3.5 (2013), pp. 1–4 (cit. on p. 26).

[48] D. Namiot and M. Sneps-Sneppe. "CAT – cars as tags". en. In: *2014 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall)*. St. Petersburg, Russia: IEEE, Oct. 2014, pp. 50–53. ISBN: 978-1-4799-5270-0. DOI: 10.1109/Nets4CarsFall.2014.7000912 (cit. on p. 7).

[49] A. Nayak. "Type of NOSQL Databases and its Comparison with Relational Databases". en. In: *International Journal of Applied Information Systems* 5 (2013), p. 4 (cit. on p. 28).

[50] Node.js. *Node.js*. en. URL: https://nodejs.org/en/ (visited on 02/05/2023) (cit. on p. 62).

[51] S. G. Obreja and A. Vulpe. "Evaluation of an Indoor Localization Solution Based on Bluetooth Low Energy Beacons". en. In: *2020 13th International Conference on Communications (COMM)*. Bucharest, Romania: IEEE, June 2020, pp. 227–231. ISBN: 978-1-72815-611-8. DOI: 10.1109/COMM48946.2020.9141987 (cit. on p. 14).

[52] F. Orujov and R. Maskeliunas. "Comparative Analysis of the Indoor Positioning Algorithms using Bluetooth Low Energy Beacons". en. In: (), p. 5 (cit. on p. 22).

[53] *Overview - Material UI*. en. URL: https://mui.com/material-ui/getting-started/overview/ (visited on 02/05/2023) (cit. on p. 55).

[54] *PDFKit*. URL: https://pdfkit.org/ (visited on 02/05/2023) (cit. on p. 63).

[55] D. Petkov and O. Petkova. "One way to make better decisions related to IT outsourcing". en. In: (1998) (cit. on p. 24).

[56] K. Phutcharoen, M. Chamchoy, and P. Supanakoon. "Accuracy Study of Indoor Positioning with Bluetooth Low Energy Beacons". en. In: *2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*. Pattaya, Thailand: IEEE, Mar. 2020, pp. 24–27. ISBN: 978-1-72816-398-7. DOI: 10.1109/ECTIDAMTNCON48261.2020.9090691 (cit. on p. 8).

[57] J. Poushter et al. "Smartphone ownership and internet usage continues to climb in emerging economies". In: *Pew research center* 22.1 (2016), pp. 1–44 (cit. on p. 23).

[58] K. Qi. *React Circular Progressbar*. original-date: 2016-05-29T20:08:10Z. Sept. 2022. URL: https://github.com/kevinsqi/react-circular-progressbar (visited on 02/05/2023) (cit. on p. 55).

[59] P. P. Ray. "An Introduction to Dew Computing: Definition, Concept and Implications". en. In: *IEEE Access* 6 (2018), pp. 723–737. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2775042 (cit. on p. 24).

[60] S. Raza et al. "Bluetooth smart: An enabling technology for the Internet of Things". en. In: *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Abu Dhabi, United Arab Emirates: IEEE, Oct. 2015, pp. 155–162. ISBN: 978-1-4673-7701-0. DOI: 10.1109/WiMOB.2015.7347955 (cit. on p. 6).

[61] *React – A JavaScript library for building user interfaces*. en. URL: https://reactjs.org/ (visited on 02/05/2023) (cit. on p. 55).

[62] *React Data Grid component - MUI X*. URL: https://mui.com/pt/x/react-data-grid/ (visited on 02/05/2023) (cit. on p. 55).

[63] *React Router: Declarative Routing for React*. en. URL: https://reactrouter.com (visited on 02/05/2023) (cit. on p. 56).

[64] *Recharts*. URL: https://recharts.org/en-US/ (visited on 02/05/2023) (cit. on p. 55).

[65] S. Sadowski and P. Spachos. "Optimization of BLE Beacon Density for RSSI-Based Indoor Localization". en. In: *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. Shanghai, China: IEEE, May 2019, pp. 1–6. ISBN: 978-1-72812-373-8. DOI: 10.1109/ICCW.2019.8756989 (cit. on pp. 14, 15).

[66] *Sass: Syntactically Awesome Style Sheets*. URL: https://sass-lang.com/ (visited on 02/05/2023) (cit. on p. 55).

[67] D. Sikeridis, I. Papapanagiotou, and M. Devetsikiotis. "BLEBeacon: A Real-Subject Trial Dataset from Mobile Bluetooth Low Energy Beacons". en. In: *arXiv:1802.08782 [cs]* (May 2019). arXiv: 1802.08782 (cit. on p. 7).

[68] M. Sneps-Sneppe and D. Namiot. "On physical web models". en. In: *2016 International Siberian Conference on Control and Communications (SIBCON)*. Moscow, Russia: IEEE, May 2016, pp. 1–6. ISBN: 978-1-4673-8383-7. DOI: 10.1109/SIBCON.2016.7491675 (cit. on p. 13).

[69] P. Spachos and K. N. Plataniotis. "BLE Beacons for Indoor Positioning at an Interactive IoT-Based Smart Museum". en. In: *IEEE Systems Journal* 14.3 (Sept. 2020), pp. 3483–3493. ISSN: 1932-8184, 1937-9234, 2373-7816. DOI: 10.1109/JSYST.2020.2969088 (cit. on p. 17).

[70] S. Subedi et al. "Beacon based indoor positioning system using weighted centroid localization approach". en. In: *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. Vienna, Austria: IEEE, July 2016, pp. 1016–1019. ISBN: 978-1-4673-9991-3. DOI: 10.1109/ICUFN.2016.7536951 (cit. on p. 20).

[71] B. S. Swamy et al. "BLE Beacon Based Museum Knowledge Sharing Platform". en. In: *NCICCNDA*. AIJR Publisher, June 2018, pp. 94–98. ISBN: 978-81-936820-0-5. DOI: 10.21467/proceedings.1.15 (cit. on p. 9).

[72] S. N. Swamy and S. R. Kota. "An Empirical Study on System Level Aspects of Internet of Things (IoT)". en. In: *IEEE Access* 8 (2020), pp. 188082–188134. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3029847 (cit. on p. 1).

[73] M. B. R. Vaja. "Retail management". In: *International Journal of Research and Analytics Reviews* 2.1 (2015), pp. 22–28 (cit. on p. 1).

[74] S. Vigneshwaran et al. "Using infrastructure-provided context filters for efficient fine-grained activity sensing". en. In: *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. St. Louis, MO, USA: IEEE, Mar. 2015, pp. 87–94. ISBN: 978-1-4799-8033-8. DOI: 10.1109/PERCOM.2015.7146513 (cit. on p. 9).

[75] *Visual Studio Code - Code Editing. Redefined*. en. URL: https://code.visualstudio.com/ (cit. on pp. 54, 61).

[76] K. Wankhede, B. Wukkadada, and V. Nadar. "Just Walk-Out Technology and its Challenges: A Case of Amazon Go". en. In: *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. Coimbatore: IEEE, July 2018, pp. 254–257. ISBN: 978-1-5386-2456-2. DOI: 10.1109/ICIRCA.2018.8597403 (cit. on p. 2).

[77] M. B. Yaakop et al. "Bluetooth 5.0 throughput comparison for internet of thing usability a survey". en. In: *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*. Kuala Lumpur: IEEE, Sept. 2017, pp. 1–6. ISBN: 978-1-5386-1805-9 978-1-5386-1807-3. DOI: 10.1109/ICE2T.2017.8215995 (cit. on p. 6).

[78] J. Yang et al. "Beyond beaconing: Emerging applications and challenges of BLE". en. In: *Ad Hoc Networks* 97 (Feb. 2020), p. 102015. ISSN: 15708705. DOI: 10.1016/j.adhoc.2019.102015 (cit. on p. 2).